Congresso de Segurança da Informação das Fatec

O uso de algoritmos de *Machine Learning* em ataques de *Phishing*

The use of Machine Learning algorithms in Phishing attacks

Henrique Minillo Moreira[1], Pedro Henrique Biazi[1], Richard Aparecido Gabriel Alves Pereira[1], Isaque Katahira[1], Faculdade de Tecnologia de Ourinhos (FATEC Ourinhos)

{henrique.moreira01,pedro.biazi}@fatec.sp.gov.br

{richard.pereira4,isaque.katahira}@fatec.sp.gov.br

Resumo

Este artigo propõe uma análise do uso de algoritmos de Machine Learning para detectar ataques de phishing em URLs. Por meio de uma análise comparativa, foi utilizado um *dataset* disponibilizado pela plataforma *Kaggle*, contendo 10.000 dados e 48 *features*. Foram aplicados três algoritmos principais: Árvore de Decisão, Support Vector Machine (SVM) e Naive Bayes (NB). Os testes mostraram que o SVM, após ajuste de parâmetros pelo módulo GridSearchCV, obteve uma acurácia de 96,48%. Contudo, todos os algoritmos mantiveram níveis de acurácia semelhantes, com SVM e Árvore de Decisão utilizando mais características do que NB para alcançar a mesma eficácia.
Palavras-chave: Machine Learning; Phishing; Árvore de Decisão; SVM; NB;

Abstract

This article proposes an analysis of the use of Machine Learning algorithms to detect phishing attacks in URLs. Through a comparative analysis, a dataset made available on the Kaggle platform was used, containing 10,000 entries and 48 *features*. Three main algorithms were applied: Decision Tree, Support Vector Machine (SVM) and Naive Bayes (NB). The tests showed that the SVM, after parameter adjustment using the GridSearchCV module, achieved an accuracy of 96.48%. Nonetheless, all algorithms maintained similar accuracy levels, with SVM and Decision Tree utilizing more features than NB to achieve the same effectiveness.
Keywords: Machine Learning; Phishing; Decision Tree; SVM; NB;

1. Introdução

A Segurança da Informação tornou-se essencial na era digital moderna, impulsionada pelo aumento exponencial de dados e pela interconexão global. A tecnologia, além de ocupar

um papel central em diversos setores, destaca-se no mercado atual como um fator de extrema relevância. No entanto, o avanço tecnológico também trouxe um aumento nos tipos e complexidade dos ataques cibernéticos, como o *phishing*. Apesar de ser um método de ataque já conhecido, o phishing continua sendo utilizado por cibercriminosos para obter vantagens pessoais, atingindo muitas vezes pessoas fora de contextos corporativos. Segundo dados da Kaspersky (2023), cerca de 82,71% dos ataques de *phishing* ocorrem pelo *WhatsApp*, sendo na maioria dos casos, sites fraudulentos.

As empresas também são alvos frequentes. De acordo com o *Data Breach Investigations Report* (DBIR) da Verizon (2023), elaborado em parceria com diversas empresas de tecnologia, cerca de 74% das violações de segurança envolvem o fator humano, com o phishing e o roubo de credenciais figurando entre os principais métodos de ataque.

De acordo com Souza et al. (2019) o combate ao phishing é feito principalmente através de sites públicos, como o *PhishTank*, onde URL's maliciosas são catalogadas e inseridas em listas de bloqueio, embora a efetividade dessa abordagem seja limitada pela dificuldade de identificar URLs suspeitas em tempo real.

Uma defesa essencial contra ataques de phishing é a detecção e mitigação de incidentes com o auxílio de *softwares* especializados. Contudo, segundo Ariza et al. (2022), a eficácia desses mecanismos é questionada, dada a sofisticação crescente dos ataques e a rápida evolução das técnicas dos invasores. Neste cenário, a criação de mecanismos de defesa mais adaptáveis e robustos torna-se imprescindível.

Este artigo busca aprofundar o conhecimento sobre algoritmos de *Machine Learning* específicos para a detecção de ataques de *phishing*, ameaças cada vez mais sofisticadas e dinâmicas no cenário cibernético. A pesquisa analisa diferentes algoritmos de *Machine Learning*, buscando não apenas identificar os métodos mais eficazes para detectar e mitigar *phishing*, mas também apontar desafios e lacunas. Com isso, o estudo fornece *insights* valiosos para profissionais de segurança cibernética e cientistas de dados, além de auxiliar organizações na tomada de decisões.

Para a análise, são utilizados conjuntos de dados disponíveis na plataforma *Kaggle* que contém 10.000 URL's e 48 *features* ligadas às características de uma URL, como por exemplo o tamanho da URL, além disso o projeto também conta com implementações em *Python*. A pesquisa envolverá análises comparativas entre ferramentas e algoritmos de classificação, sendo eles: Árvore de Decisão, *Support Vector Machine* (SVM) e *Naive*

Bayes. Para medidas de avaliação são utilizadas métricas tradicionais como acurácia, precisão, revocação e *F1-Score*.

O objetivo geral deste artigo é investigar o uso da inteligência artificial(IA) na criação de algoritmos de detecção de *phishing* e comparar a eficácia de diferentes algoritmos de *Machine Learning* em relação a ataques de *phishing*. Para concluir o projeto, destaca-se os seguintes objetivos específicos:

1. Pré-processamento do conjunto de dados;
2. Analisar *features* relevantes;
3. Comparar a eficiência dos diferentes algoritmos;

Na seção seguinte é apresentado o referencial teórico necessário para a construção do artigo.

2. Referencial Teórico

Nesta seção, são apresentados os diferentes termos e tecnologias a serem utilizados para alcançar os objetivos deste artigo. O conteúdo abordado nas seguintes subseções é relacionado a temas associados ao projeto, como engenharia social, inteligência artificial e aprendizado de máquina.

2.1. *Phishing*

Segundo a Cartilha de Segurança da *Internet*, disponibilizado pela CERT.br (2012), *phishing* é um tipo de fraude que utiliza técnicas de engenharia social para atrair a atenção dos usuários. O ataque normalmente ocorre por meio de envio de *e-mails* falsos, onde os golpistas tentam se passar por pessoas, empresas, serviços, assim induzindo o usuário a fornecer informações pessoais e muitas vezes confidenciais. Da mesma forma, nota-se que CERT.br (2012) alerta sobre o *pharming*, que é um tipo mais específico de *phishing*, no qual o usuário faz uma tentativa de acessar um *site* legítimo, porém o seu navegador *Web* é redirecionado para uma página falsa criada pelo golpista, para tal, o atacante utiliza códigos maliciosos que alteram configurações de DNS (*Domain Name System*).

O autor apresenta meios de prevenção contra esse tipo de fraude, por exemplo, a utilização de mecanismos de segurança como *antimalwares* e filtros *antiphishing*, bem como a educação e conscientização sobre o uso seguro da *internet*. Além disso, autores como Barros, Silva e Miranda (2020) e Resnick e Bastos-Filho (2023), defendem que a

inteligência artificial e o aprendizado de máquina são dois mecanismos eficazes para a detecção de ataques de *phishing*.

2.2. *Machine learning*

Do inglês *Machine learning*, o aprendizado de máquina é uma das áreas da inteligência artificial que segundo COUTINHO (2023), abrange o desenvolvimento de sistemas que consigam aprender de forma autônoma, identificar padrões e tomar decisões por meio de uma grande quantidade de dados. O aprendizado de máquina é usado em uma variedade de setores, incluindo negócios, saúde, finanças, educação e entretenimento. Ele pode ser usado para prever o comportamento do consumidor, identificar fraudes, identificar condições médicas, fazer recomendações de produtos e conteúdos, entre outras coisas. A tecnologia permite que as máquinas se adaptem e melhorem com o tempo, tornando-se cada vez mais precisas e eficientes.

É possível classificar o aprendizado de máquina em três diferentes tipos, conforme Raschka e Mirjalili (2018):

- **Aprendizado supervisionado:** Que utiliza dados rotulados para o treinamento de um algoritmo que possa prever dados desconhecidos e dados futuros. O termo supervisionado deriva-se do fato dos dados de treinamento já serem rotulados;
- **Aprendizado de reforço:** Um sistema que age de acordo com seu ambiente para atingir uma determinada recompensa. Um exemplo citado pelo autor é um jogo de xadrez onde o algoritmo age de acordo com o estado do tabuleiro(ambiente) e a vitória ou derrota seria sua recompensa;
- **Aprendizado não-supervisionado:** Um sistema não-supervisionado é aquele onde os dados são desconhecidos, não possuem rótulos, nem uma váriavel de recompensa. Esse algoritmo tem o objetivo de encontrar padrões e informações dentro destes dados.

2.3. Medidas de avaliação de classificador

Tratando-se de *Machine learning*, existem diferentes tipos de métodos de avaliar a qualidade de classificações de um modelo, contudo para este trabalho serão considerados métodos tradicionais como a matriz de confusão e suas respectivas características.

Conforme Souza (2023), ao avaliar os resultados da classificação de algum modelo, é possível dividí-los em quatro categorias diferentes, especificamente:

- **Verdadeiro Positivo (TP):** Uma predição de amostra positiva que é de fato positiva;
- **Verdadeiro Negativo (TN):** Uma predição de amostra negativa que é de fato negativa;
- **Falso Positivo (FP):** Uma predição de amostra positiva que é, na verdade, negativa;
- **Falso Negativo (FN):** Uma predição de amostra negativa que é, na verdade, positiva.

Segundo Tang e Mahmoud (2021), os casos em que as amostras retornam uma predição de falso negativo (FN) podem ser extremamente danosas, uma vez que os ataques não são detectados pelo modelo. Portanto, existem métricas para medir a qualidade das predições do modelo. Ainda de acordo com Tang e Mahmoud (2021), existem diferentes modelos de métrica.

A acurácia é a métrica utilizada para medir a qualidade geral das predições. Sendo ela, a mais comum entre elas, a acurácia quantifica a porcentagem de acerto geral do modelo de acordo com as predições corretas em relação à quantidade total de predições. Na equação 1 é possível visualizar como é realizado o cálculo da acurácia:

$$\textbf{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

O autor destaca o fato de que nem sempre um modelo com uma acurácia alta é um modelo excelente, por exemplo, em um modelo de 90% de acurácia, a depender da quantidade de dados, muitos ataques podem passar despercebidos, por isso a importância de outras métricas como a precisão. A precisão é a porcentagem dos dados TP em relação à todos os dados identificados corretamente que são positivos. Constata-se na seguinte equação 2 a representação matemática da precisão:

$$\textbf{Precisão} = \frac{TP}{TP + FP} \tag{2}$$

A revocação, do inglês *Recall*, é a porcentagem de dados rotulados como TP em relação aos dados identificados como positivos corretamente e os dados falsos negativos. Essa equação é importante para avaliar a eficácia do modelo em relação à ataques não detectados. Na equação 3, nota-se a representação matemática da revocação:

$$\textbf{Revocação} = \frac{TP}{TP + FN} \tag{3}$$

Ademais, existe também o *F1-Score* ou apenas *F-Score*, que é a combinação da precisão e revocação. Na seguinte equação 4, constata-se a representação matemática do *F1-Score*:

$$\textbf{\textit{F1-Score}} = \frac{2 * \textbf{Precisão} * \textbf{Revocação}}{\textbf{Precisão} + \textbf{Revocação}} \tag{4}$$

O *F1-Score* é utilizado para avaliar o equilíbrio geral do algoritmo, quando maior o *F1-Score*, maiores são as taxas de precisão e revocação, o que indica um modelo de alta qualidade.

2.4. Algoritmo de Árvore de decisão

O algoritmo de Árvore de decisão é um algoritmo de *Machine Learning* que, segundo Russell e Norvig (2009), representa uma função que recebe um vetor de valores de atributos como entrada e retorna uma decisão, sendo esse um valor de saída único. Dentro do contexto do trabalho acaba sendo uma escolha interessante, uma vez que, os algoritmos tentarão identificar se uma entrada é ou não um ataque de *phishing*.

O algoritmo funciona a partir de uma sequência de vários testes, onde a quantidade de *features* de um modelo influencia diretamente na quantidade de testes que o algoritmo terá de fazer para tomar uma decisão, Russell e Norvig (2009) ainda afirma que a árvore de decisão é um algoritmo relativamente simples, inclusive usado no dia a dia na tomada de decisões, porém não deixa de ser um algoritmo muito útil como um algoritmo de classificação .

2.5. Algoritmo *Support Vector Machine (SVM)*

O algoritmo de *Support Vector Machine* (SVM) é atualmente um dos algoritmos mais populares quando se trata de aprendizagem supervisionada, principalmente pela sua flexibilidade em lidar com diferentes tipos de dados, segundo Patle e Chouhan (2013) ele tem fortes propriedades regularizadoras para a generalização de dados por ser um algoritmo analisa um conjunto de dados em diferentes dimensões, e por isso, acaba sendo muito útil para lidar com casos em que um grupo de dados não possam ser completamente separados, também é um algoritmo flexível, podendo ser usado tanto para algoritmos de

classificação ou regressão.

Do mesmo modo, Patle e Chouhan (2013) afirma que o SVM possui quatro conceitos básicos, sendo eles:

- **Hiperplano separador**: O hiperplano seria um termo geral para uma linha reta em um espaço de alta dimensão. O hiperplano separador é aquele que separa as diferentes amostras de um conjunto de dados;
- **Hiperplano de margem máxima**: O hiperplano de margem máxima é a seleção da linha que separa as duas classes, porém aquela que possui a maior distância entre qualquer uma das amostras;
- **Margem suave**: A margem suave serve para garantir que o algoritmo funcione sem muitos problemas, fazendo com que dados anômalos atravessem uma margem, de uma forma, que não afete o resultado final;
- **Função de *Kernel***: As funções de *kernel* são algoritmos matemáticos que permitem o SVM em realizar uma projeção de dados de um espaço de baixa dimensão em um espaço de dimensão maior, permitindo uma análise mais precisa daqueles dados, existem diferentes funções de *kernel*, dentre as citadas pelo autor encontram-se: Função *Kernel* Polinomial, Função *Kernel* Linear, Função de Base Radial e Função Sigmoide.

De acordo com Russell e Norvig (2009), existem três principais propriedades que tornam o algoritmo SVM altamente requisitado. A primeira delas é o separador de margem máxima, que atua como um limite de decisão, ajudando na generalização dos dados. Em segundo lugar, os SVMs não se limitam a métodos estritamente lineares. Embora normalmente criem uma separação linear em um hiperplano, é possível ajustar esse comportamento para trabalhar com dados em um plano superior, o que facilita o processamento de dados não lineares. Por fim, os SVMs são algoritmos não paramétricos, o que significa que eles armazenam os exemplos de treinamento apresentados, mas na prática mantêm apenas aqueles que são relevantes para definir a separação entre os exemplos, evitando assim a superadaptação.

2.6. Algoritmo de *Naive Bayes*

Sendo um dos algoritmos de classificação mais simples e conhecidos, o algoritmo de *Naive Bayes* baseia-se, segundo Chen et al. (2020), na possibilidade de um exemplo novo pertencer a alguma classe já conhecida dentro dos dados de treinamento, funcionando

quase que como um ”adivinhador”. Por exemplo, em um contexto de festa, existem pistas que indicam tipos de doces diferentes, como formato, cor, sabor, entre outros.

Conforme novas sobremesas vão chegando, o algoritmo tenta adivinhar o tipo da sobremesa baseado naquelas que já estavam dentro de seus dados, supondo que uma nova sobremesa tem um formato redondo, possui sabor doce e cor marrom, o algoritmo pode supor que a sobremesa seja um bolo. Esse é um exemplo de algoritmo de *Naive Bayes* ”ingênuo”, sendo este, o modelo mais comum de rede *bayesiana* utilizado em aprendizagem de máquina.

Como descrito por Russell e Norvig (2009), o modelo é chamado de ingênuo porque os atributos são dependentes condicionalmente uns dos outros. Porém o autor afirma que o algoritmo, apesar de não aprender tanto quanto a árvore de decisão, é muito amplo e se adapta muito bem em relação a grandes quantidades de dados e dados perdidos.

2.7. Trabalhos relacionados

No âmbito acadêmico vêm sendo desenvolvidos diversos trabalhos que visam a detecção de ataques de *phishing* utilizando diversos algoritmos e *datasets* diferentes. Nesta subseção serão destacados alguns trabalhos que possuem um certo nível de similaridade quanto à descoberta da eficiência de diversos algoritmos na detecção de *phishing*.

O trabalho de Tang e Mahmoud (2021) de aprendizado de máquina aplicados à detecção de *phishing* inclui 7 algoritmos diferentes, geralmente em abordagens de classificação supervisionada. Métodos híbridos, como combinações de *extra-trees* com *Adaboost* e *Bagging*, destacam-se por reduzir falsos positivos e melhorar a acurácia. Para otimização, técnicas de seleção de *features*, como RFE (*Recursive Feature Elimination*), IG (*Information Gain*) e PCA (*Principal Components Analysis*), ajudam a refinar o *dataset*, acelerando o processamento e prevenindo *overfitting*. O autor utiliza diversos *datasets* de diversas fontes diferentes, fortificando ainda mais os resultados de seus testes.

Analogamente, Resnick e Bastos-Filho (2023) aplica três diferentes algoritmos, sendo eles: Floresta Aleatória, *XGBoost* e Rede Neural MLP. Os autores também usam uma abordagem de aprendizagem supervisionada, utilizando um *dataset* de cerca de 50.000 dados de entrada e contendo um total de 15 *features*. Após o pré-processamento e análise dos dados, os autores otimizaram os parâmetros de cada modelo, conseguindo resultados expressivos, com o algoritmo de Floresta Aleatória conseguindo 100% de acurácia, precisão, revocação e *F1-score*.

Por fim, COUTINHO (2023) utiliza também, do mesmo modo, uma abordagem de aprendizagem supervisionada, desta vez, utilizando quatro modelos: Árvore de decisão, Floresta Aleatória, *Extra-trees* e *XGBoost*. O autor utiliza um *dataset* contendo 88.647 entradas de dados com cerca de 112 *features*. Destaca-se o uso da plataforma SHAP para analisar a relevância de cada *feature* e também o resultado obtido pelo autor, onde o algoritmo de *XGBoost* conseguiu uma acurácia de 97,44%, sendo essa a maior do projeto.

Apesar de similares, denota-se o uso de diferentes modelos estudados em cada trabalho, bem como os diferentes métodos e *datasets* utilizados por cada autor.

3. Metodologia

Este estudo examina a Segurança da Informação por meio de uma revisão da literatura. Foram realizadas buscas no Google Acadêmico utilizando palavras-chave como ”*phishing*”, ”*machine learning*”e ”inteligência artificial”. As buscas abrangeram dados e pesquisas de 2010 até junho de 2024. Nesta seção, são apresentados os materiais e métodos utilizados para a continuidade do projeto.

Esse método de pesquisa pode ser descrito com base nas contribuições de autores proeminentes da área acadêmica. Ao considerar as abordagens propostas por esses especialistas, é possível delinear um caminho claro para a investigação e solução de problemas complexos. Desde a revisão da literatura, conforme delineada por Webster e Watson (2002), até a análise comparativa, como destacado por Yin (2014), este método se baseia em uma base sólida de conhecimento prévio. Ao incorporar técnicas específicas, como a detecção de *phishing*, o modelo proposto segue a orientação de autores como Creswell (2014), que defendem a integração de métodos quantitativos e qualitativos para uma compreensão abrangente do fenômeno em questão. Assim, essa abordagem oferece uma estrutura robusta para a pesquisa, alinhada com as melhores práticas e os mais recentes avanços na área de estudo.

Portanto, o método descrito pode ser entendido como uma síntese entre a revisão de literatura, que proporciona uma base teórica sólida, a análise comparativa, que permite a comparação entre diferentes abordagens ou ferramentas, e a pesquisa quantitativa, que utiliza métodos estatísticos para investigar relações e tendências em uma população específica. As medidas de avaliação utilizadas são as métricas acurácia, precisão, revocação e *F1-score*, que foram citadas na Revisão de Literatura 2. A seguir são apresentados os métodos utilizados para a comparação dos algoritmos e dos resultados.

3.1. *Dataset*

Para realizar os testes práticos, são utilizados conjuntos de dados disponibilizados pela plataforma *Kaggle*, que é uma plataforma criada em 2010 voltada para a área de Ciência de Dados e *Machine Learning*, dentro dela estão disponíveis diversos conjuntos de dados elaborados pela própria comunidade de cientistas de dados e desenvolvedores. Para este artigo, é utilizado um *dataset* específico de ataques de *phishing*.

O *dataset* que será utilizado foi elaborado por Tan (2018) e está disponibilizado na plataforma *Kaggle*, este conjunto de dados possui um total de 10.000 entradas, sendo 5.000 delas URL's *phishing*, e as outras 5.000 sendo URL's legítimas. As URL's de *phishing* foram extraídas de bancos de dados como o *PhishTank* e *OpenPhish* e as URL's legítimas foram extraídas de sites como *Alexa* e *Common Crawl*.

O conjunto de dados oferece um total de 48 *features*. Entre elas estão algumas características como o tamanho da URL, número de pontos na URL, se a URL possui o protocolo HTTPS (*Hypertext Transfer Protocol Secure*), entre outros, sendo 1 o número que classifica uma URL maligna, e 0 uma URL benigna.

3.2. *Google Colab*

O *Google Colab*, também chamado de *Colaboratory*, é uma plataforma gratuita de computação em nuvem oferecida pelo *Google*. Com ela, é possível acessar um ambiente interativo pela nuvem, possibilitando a criação e execução de códigos, sem utilizar o processamento do computador físico do usuário, utilizando apenas o serviço em nuvem.

3.3. *Python*

Em relação à produção dos algoritmos, é utilizada a linguagem de programação *Python*, sendo uma linguagem muito ampla e também muito flexível para lidar com algoritmos de *Machine Learning*, isso é por causa de suas diversas bibliotecas que permitem a manipulação de diversos dados e é muito completa, tanto na parte do pré-processamento dos dados, quanto do treinamento de um algoritmo. Entre as bibliotecas utilizadas para a criação dos algoritmos, estão:

- ***Pandas:*** Biblioteca utilizada para manipulação de dados, permitindo uma análise e filtragem de dados de maneira otimizada;
- ***Numpy:*** É uma biblioteca recomendada para lidar com arrays e matrizes de dados, além de fornecer funções matemáticas eficientes. O *NumPy* permite fazer

operações em grandes quantidades de dados de forma rápida, já que é otimizado para trabalhar com números;

- ***Scikit-learn:*** Biblioteca voltada para machine learning, possibilitando a criação de algoritmos de classificação e regressão, além de diversas funções de pré-processamento;
- ***Matplotlib e Seaborn:*** Ambas são bibliotecas de visualização e análise de dados, ótimas para gerar gráficos e analisar informações.

Para mitigar desafios de classificação, como o excesso de adaptação aos dados (*overfitting*) ou a dificuldade em capturar as tendências dos dados (*underfitting*), uma abordagem eficaz é a aplicação da técnica de *cross-validation*, que foi utilizada para a obtenção de melhores resultados. A figura 1 a seguir representa um diagrama de etapas que aborda os processos que são feitos para a conclusão do projeto:

Figura 1. Diagrama de etapas.

Fonte: Elaborado pelo autor

Primeiramente é feito um pré-processamento do conjunto de dados, o método utilizado é a normalização *Min-Max*, onde o conjunto é transformado em uma escala de 0 a 1, de forma que os dados possam ser mais fáceis de comparar, além de ajudar na análise de variabilidade dos dados, após isso são elaborados os algoritmos de classificação, em seguida, são realizados os testes práticos em busca das métricas de cada algoritmo classificador, bem como a comparação da eficácia dos diferentes algoritmos, e por fim a compilação dos resultados apresentados na seção seguinte. Os códigos utilizados durante o projeto podem ser acessados pela plataforma do *GitHub* através do seguinte *link*: ⟨https://github.com/Hnrqmeans/trabalho_de_graduacao⟩.

4. Resultados e Discussões

Os testes práticos são realizados na plataforma do *Google Colab*, primeiramente, desenvolveram-se testes com a finalidade de encontrar *features* que melhor identificariam

um ataque de *phishing*. Os algoritmos estudados são os de Árvore de Decisão, usando o módulo ”*tree*”do *Sklearn*, bem como o *Support Vector Machine* (SVM) utilizando o módulo ”svm”do *Sklearn*, e por último o *BernoulliNB* sendo ele um dos diferentes tipos de algoritmos baseados no *Nayve Bayes*, e também uma boa opção para modelos de classificação binária.

A partir dos módulos *SelectKBest* e *MutualInfClassif* do *Scikit-Learn*, é possível destacar como as *features* funcionam em cada algoritmo, como mencionado na tabela 1:

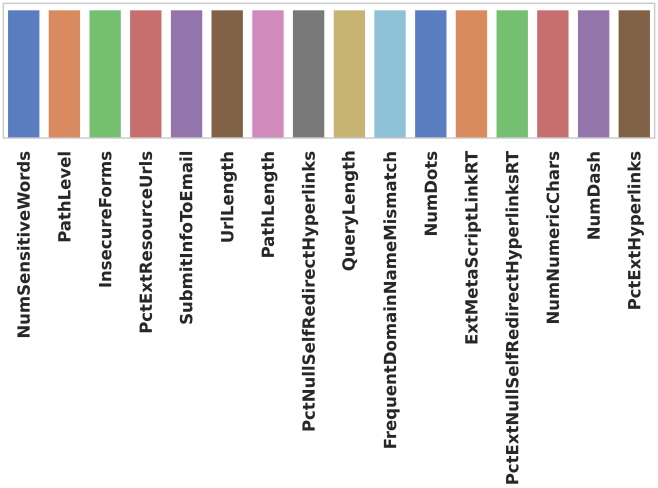
Tabela 1. Relação entre os modelos e as *features*.

Modelo	**Valor de K (*Features*)**
Árvore de decisão	30
SVM	33
BernoulliNB	19

Fonte: Elaborado pelo Autor.

Na tabela acima, denota-se que cada algoritmo responde melhor com um número de *features* diferentes, o módulo *SelectKBest* encontra as melhores *features* dentro do algoritmo selecionado, porém algumas *features* se destacam, como representado na figura 2 a seguir:

Figura 2. Valores comuns de *features* em ambos os modelos.



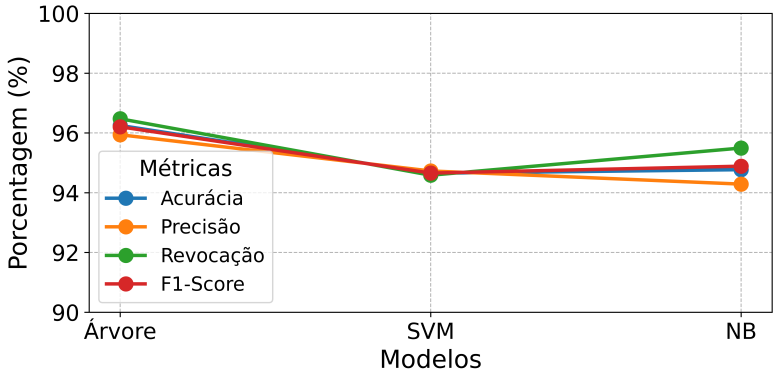
Fonte: Elaborado pelo autor

Mesmo dentre as diferentes particularidades de cada algoritmo, as *features* mostradas na figura 2 apareceram em ambos os algoritmos, sendo possivelmente as mais

relevantes dentro do conjunto de dados utilizado.

Logo após, deu-se início aos testes de eficiência dos diferentes modelos, sobretudo, os testes aconteceram aplicando o método de *K-fold cross-validation* para uma melhor avaliação do conjunto de dados. Do mesmo modo, é importante destacar que 80% dos dados são utilizados para o treinamento e 20% para a realização dos testes, empregando o número de 5-*folds* para teste. A figura 3 demonstra os resultados dos testes com os parâmetros padrão dos algoritmos:

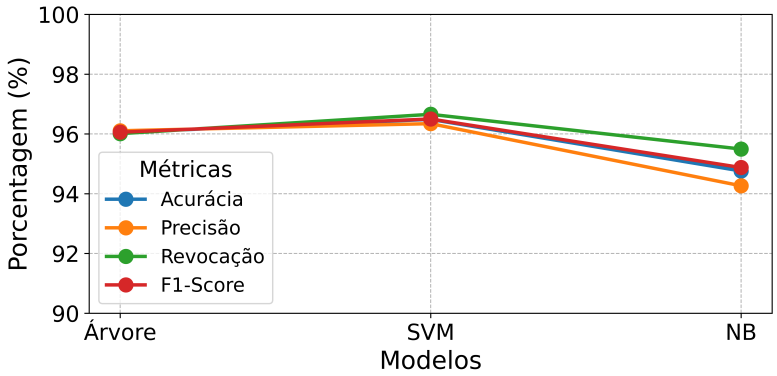
Figura 3. Desempenho dos modelos sem alterações.



Fonte: Elaborado pelo autor.

Do mesmo modo, são realizados testes em busca de parâmetros que explorassem uma melhora na eficiência de cada algoritmo, para isso, foi adotado o módulo *GridSearchCV* da biblioteca do *Scikit-learn*, que testa um determinado algoritmo e tenta encontrar os melhores parâmetros para ele. A figura 4 representa os resultados dos testes após os parâmetros encontrados com o *GridSearchCV*:

Figura 4. Desempenho dos modelos após parâmetros do *GridSearchCV*.



Fonte: Elaborado pelo autor.

Após a aplicação dos parâmetros, houve uma notável melhora no algoritmo de SVM em relação à classificação de um ataque de *phishing*, enquanto os outros modelos mantiveram métricas similares ou até mesmo um pouco inferiores. Para uma melhor visualização, a tabela 2 aborda as métricas de avaliação de cada algoritmo:

Tabela 2. Resultados dos testes.

Modelo	Acurácia	Precisão	Revocação	*F1-Score*
Árvore de decisão	96.25	95.93	96.47	96.20
SVM	94.65	94.72	94.58	94.65
Bernoulli NB	94.77	94.28	95.49	94.88
Árvore de decisão*	96.05*	96.10*	96.00*	96.05*
SVM*	96.48*	96.34*	96.65*	96.50*
*Bernoulli NB**	94.76*	94.26*	95.49*	94.87*

Fonte: Elaborado pelo Autor.
**Métricas após os parâmetros do GridSearch.*

Como observado, apesar do SVM ter tido uma melhora em sua eficiência, não foi uma mudança muito grande, isso também é válido para os outros algoritmos que não apresentaram mudanças significantes em suas métricas além do número de *features* utilizados.

A pesquisa analisa os diferentes comportamentos de algoritmos de *Machine Learning* na detecção de URL's *phishing*, e também aborda sobre possíveis *features* que são importantes na identificação desse ataque. A partir de um conjunto de dados com 10.000 URLs, metade delas maliciosas e metade legítimas, após o pré-processamento dos dados com normalização *Min-Max*, são aplicados os algoritmos de Árvore de Decisão, *Support Vector Machine* (SVM) e *Naive Bayes*. Cada modelo teve um número otimizado de características: 30 para Árvore de Decisão, 33 para SVM e 19 para *Naive Bayes*.

Os testes são realizados no *Google Colab*, com o apoio de bibliotecas como *Pandas*, *Numpy*, *Matplotlib* e *Seaborn* para visualização dos resultados e análise estatística. Além disso, é utilizado o módulo *GridSearchCV* da *Scikit-learn*, que testa múltiplos parâmetros e identifica a combinação mais eficiente para cada algoritmo. Após a otimização, o SVM destacou-se com uma melhoria na sua eficiência embora não muito significativa, enquanto os outros dois mantiveram resultados similares. Não só isso, destaca-se que o SVM e a Árvore de decisão tiveram de utilizar mais *features* para atingir

esses resultados do que o algoritmo de *Naive Bayes*.

5. Conclusão

Esta pesquisa aborda o crescente avanço do *Machine Learning* na detecção de ataques de *phishing*, e como sua implementação pode ser muito benéfica para melhorar a segurança de uma organização. O estudo demonstra na prática a eficácia de algoritmos de *Machine Learning* para a construção de um sistema seguro de detecção de *phishing*.

O objetivo geral do artigo é investigar o uso da inteligência artificial (IA) na criação de algoritmos de detecção de *phishing* e comparar a eficácia de diferentes algoritmos de *Machine Learning* em relação a ataques de *phishing*. Os resultados mostram a eficiência dos diferentes algoritmos estudados e também aborda processos necessários para a criação dos algoritmos, como o pré-processamento.

Os objetivos específicos foram igualmente atingidos através da programação do pré-processamento, a análise das *features* mais utilizadas e também pela comparação dos resultados onde o SVM saiu levemente superior aos outros algoritmos, ao mesmo tempo que abre um grande leque para melhora contínua dos *scripts*.

Para trabalhos futuros, o estudo sugere a expansão da análise para outros tipos de ataques de engenharia social e a adaptação dos algoritmos para cenários de *Big Data* e computação quântica, onde o volume de dados e a complexidade das ameaças são ainda maiores, bem como estudos com outros algoritmos de *Machine Learning*. Da mesma forma, é possível reestruturar os códigos utilizados para que outros pesquisadores possam testar os algoritmos com diferentes conjunto de dados e também implementar mudanças para a construção de um algoritmo mais eficiente na detecção de ataques de *phishing*.

Referências

ARIZA, M. et al. Ataques automatizados de engenharia social com o uso de bots em redes sociais profissionais. In: *Anais do XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. Porto Alegre, RS, Brasil: SBC, 2022. p. 153–166. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbseg/article/view/21665⟩.

BARROS, M.; SILVA, C.; MIRANDA, P. Xphide: Um sistema especialista para a detecção de phishing. In: *Anais do XX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Porto Alegre, RS, Brasil: SBC, 2020. p. 161–174. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbseg/article/view/19235⟩.

CERT.BR. *Cartilha de Segurança para Internet*. [S.l.]: Comitê Gestor da Internet no Brasil, 2012.

CHEN, S. et al. A novel selective naïve bayes algorithm. *Knowledge-Based Systems*, v. 192, p. 105361, 2020. ISSN 0950-7051. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0950705119306185⟩.

COUTINHO, V. M. Detecção de páginas de phishing utilizando aprendizado de máquina. 2023.

CRESWELL, J. W. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. [S.l.]: SAGE Publications, 2014.

KASPERSKY. *Brasil é o país com mais ataques de phishing por WhatsApp no mundo em 2022, aponta Kaspersky*. 2023. Https://www.kaspersky.com.br/about/press-releases/2023_brasil-e-o-pais-com-mais-ataques-de-phishing-por-whatsapp-no-mundo-em-2022-aponta-kaspersky. Acesso em: 21/04/2024.

PATLE, A.; CHOUHAN, D. S. Svm kernel functions for classification. In: IEEE. *2013 International conference on advances in technology and engineering (ICATE)*. [S.l.], 2013. p. 1–9.

RASCHKA, S.; MIRJALILI, V. *Python machine learning* -. 2. ed. Birmingham, England: Packt Publishing, 2018.

RESNICK, N. E.; BASTOS-FILHO, C. J. A. Aplicação de aprendizado de máquinas para detecção de urls phishing. *Revista de Engenharia e Pesquisa Aplicada*, Revista de Engenharia e Pesquisa Aplicada, v. 9, n. 1, p. 41–49, dez. 2023. ISSN 2525-4251. Disponível em: ⟨http://dx.doi.org/10.25286/repa.v9i1.2773⟩.

RUSSELL, S.; NORVIG, P. *Artificial intelligence*. 3. ed. Upper Saddle River, NJ: Pearson, 2009.

SOUZA, C. et al. Phishkiller: Uma ferramenta para detecçao e mitigaçao de ataques de phishing através de técnicas de deep learning. In: SBC. *Anais Estendidos do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2019. p. 81–90.

SOUZA, V. S. S. Introdução à interpretabilidade de redes neurais convolucionais. 2023.

TAN, C. L. *Phishing Dataset for Machine Learning: Feature Evaluation*. Mendeley, 2018. Disponível em: ⟨https://data.mendeley.com/datasets/h3cgnj8hft/1⟩.

TANG, L.; MAHMOUD, Q. H. A survey of machine learning-based solutions for phishing website detection. *Machine Learning and Knowledge Extraction*, MDPI AG, v. 3, n. 3, p. 672–694, ago. 2021. ISSN 2504-4990. Disponível em: ⟨http://dx.doi.org/10.3390/make3030034⟩.

VERIZON. *Data Breach Investigations Report*. 2023. Https://www.verizon.com/business/resources/T86f/reports/2023-data-breach-investigations-report-dbir.pdf. Acesso em: 08/11/2023.

WEBSTER, J.; WATSON, R. T. Analyzing the past to prepare for the future: Writing a literature review. *Management Information Systems Quarterly*, v. 26, n. 2, p. 3–13, 2002.

YIN, R. K. *Case Study Research: Design and Methods*. [S.l.]: SAGE Publications, 2014.