Construção de modelo de Machine Learning para Detecção de Domínios Maliciosos

Abner F. B. Costa[1], **Michele Venturin**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

afbcosta@inf.ufpr.br, mventurin@inf.ufpr.br

Resumo. *Este artigo tem como objetivo apresentar o relatório final do projeto prático da disciplina Ciência de Dados para Segurança, ofertada pela Universidade Federal do Paraná (UFPR) e lecionada pelo professor doutor André Ricardo Abed Grégio. Este relatório tem como objetivo apresentar os resultados obtidos para o problema de classificação de nomes de domínios entre benignos e maliciosos. Nele consta uma breve introdução ao conjunto de dados e sua distribuição, atributos e características geradas, algoritmos utilizados e resultados obtidos.*

1. Introdução

Deste sua criação a Internet vem sendo regulamentada na tentativa de torná-la mais segura, porém o número de tentativas maliciosas vem crescendo verrinosamente. Muitas tentativas maliciosas são realizadas através de endereços de domínios que podem atuar das mais variadas maneiras, tais como o download automático de *malware* para o computador do usuário, solicitação e coleta de informações pessoais através da imitação de sites famosos ou ainda sendo fonte de propagação de *spam*. Desta forma, o principal propósito deste trabalho foi buscar classificar nomes de domínios sem utilizar a abordagem comum das listas negras, as quais tem dificuldade para listar os domínios maliciosos novos, que são criados a cada dia.

O objetivo foi construir um modelo capaz de classificar nomes de domínios em benigno ou malicioso, utilizado um conjunto de nomes de domínios [Mahdavifar et al. 2021] previamente rotulados como *malware*, *phishing*, *spam* e benignos.

Foi considerado também o uso de uma lista branca, onde domínios já conhecidos são reconhecidos previamente ao uso do modelo e os domínios não presentes passam então a serem analisados pelo classificador. O modelo, ao fim, retorna o domínio classificado como benigno ou maligno.

2. Dataset

Nesta seção estão apresentadas a visão geral do conjunto de dados e sua distribuição, além dos atributos e características gerados para o modelo. Os atributos e características precisaram passar por alterações ao longo da elaboração do projeto, com o propósito de alcançar a melhor eficácia para os resultados do modelo proposto.

2.1. Distribuição dos Dados

O conjunto de dados coletado tem majoritariamente domínios benignos, sendo 988299 benignos e 36709 maliciosos, como mostra a Figura 1. A Figura 2 mostra a distribuição dos domínios maliciosos do conjunto coletado. A Tabela 1 contém alguns exemplos dos dados de domínios, benignos e maliciosos, que serão representados na figuras mencionadas.

Domínio	Classificação	Classificação Binária
stockholm.se	Benigno	Benigno
arrow.com	Benigno	Benigno
ahackaday.io	Benigno	Benigno
businessbattle.tk	Malware	Malicioso
paypal.de.daten.sicherheit-benutzer.top	Phishing	Malicioso
k-slee.com	Spam	Malicioso

Table 1. Exemplos de domínios

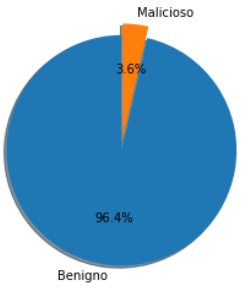

Figure 1. Distribuição do conjunto de dados coletado, com domínios classificados em benignos e malignos.

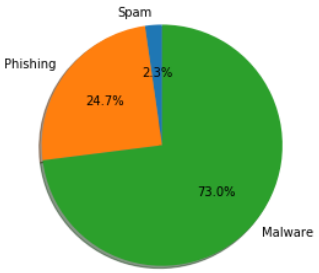

Figure 2. Distribuição do conjunto de dados coletado, com domínios maliciosos classificados em *spam*, *phishing* e *malware*.

Considerando o uso da lista branca para identificar previamente grande parte dos domínios benignos, o conjunto de dados coletado foi balanceado e o número de classes benignas foi reduzida ao mesmo número das classes maliciosas, como mostram as Figuras 3 e 4 dos dados separados para treino, e as Figuras 5 e 6, dos dados separados para teste. Os conjuntos de treino e teste foram separados em conjunto de treino compondo 80% dos dados e conjunto de teste compondo 20% dos dados.

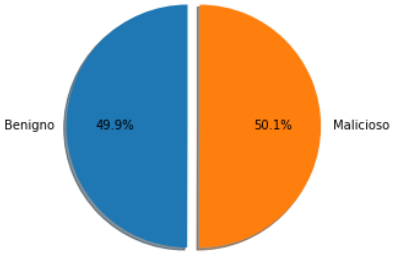

Figure 3. Domínios do conjunto de treino, classificadas entre benignos e malignos.

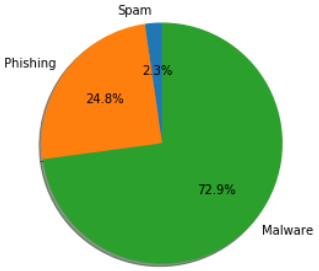

Figure 4. Domínios maliciosos do conjunto de treino, classificados entre *spam*, *phishing* e *malware*.

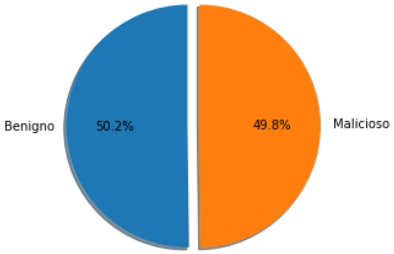

Figure 5. Conjunto de teste com domínios classificados em benignos e malignos.

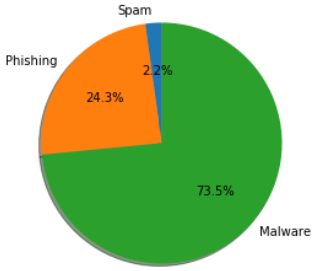

Figure 6. Domínios maliciosos do conjunto de teste, classificados entre *spam*, *phishing* e *malware*.

2.2. Atributos e Características

Para a definição dos atributos e características foram pesquisados ([Davison et al. 2019], [Bezerra et al. 2015], [Olivo et al. 2010], e [Mahdavifar et al. 2021]) indica que "1 em 4 URLs maliciosas são encontradas hospedadas em domínios confiáveis". Isto demonstra a dificuldade na identificação correta dos domínios maliciosos, porém corrobora com a ideia de que o uso de nomes de domínios muito conhecidos recebem pequenas modificações lexicais para enganar o usuário.

Utilizando as bibliotecas *IPy* e *tld* para Python3, foram gerados a partir dos domínios os seguintes atributos:

- Formato IP ou não IP.
- (SSD) *Subdomain+Second-Level Domain*
- (SUB) *Subdomain*
- (SLD) *Second-Level Domain*
- (TLD) *Top-Level Domain*

Para casos de domínio com formato de número IP em vez de texto, foram definidos valores padrão para os atributos SSD, SUB, SLD e TDL. Para casos onde a biblioteca *tld*

não foi capaz de realizar a separação do domínio, este foi atribuído na íntegra ao atributo SLD.

Através dos atributos foram geradas 58 características, dividas em características léxicas e características de terceiros. Esta última foi obtidas através das bibliotecas *whois*, *ipwhois* e do ranking Alexa[Amazon 2021]. Das 58 características, 45 são características léxicas e 13 são características de terceiros. Todas as características estão listadas no Apêndice A deste trabalho.

Como SLD famosos foram definidos os 1000 domínios mais famosos no ranking Alexa, e como métrica de distância foi utilizada a distância de Levenshtein. Para a padronização dos dados foi utilizada a função de normalização da biblioteca *sklearn* e para codificar os características de tipo *Char* foi utilizada uma codificação padrão utilizando a função *LabelEncoder*. Os dados textuais de TLD foram convertidos usando TF-IDF.

3. Algoritmos Utilizados

Como classificadores, foram escolhidos e testados 3 algoritmos: Random Forest, K-Nearest Neighbors (KNN) e Perceptron Multicamadas (MLP). Os algoritmos estão discutidos nesta seção, juntamente com os parâmetros utilizados para cada um deles.

Nesta seção também serão realizadas análises de limiar, curvas de *Precision* e *Recall*, que foram inspiradas em [Ceschin et al. 2019].

3.1. Algoritmo Random Forest

O primeiro algoritmo testado foi o algoritmo Florestas Randômicas (*RandomForesClassifier*). Para a definição do limiar da classificação foram analisadas as Figuras 7 e 8. Uma vez que o algoritmo será utilizando juntamente com uma lista branca, buscou-se um balanceamento entre métricas de *Precision* e *Recall*, estipulando assim um limiar de 0.435.

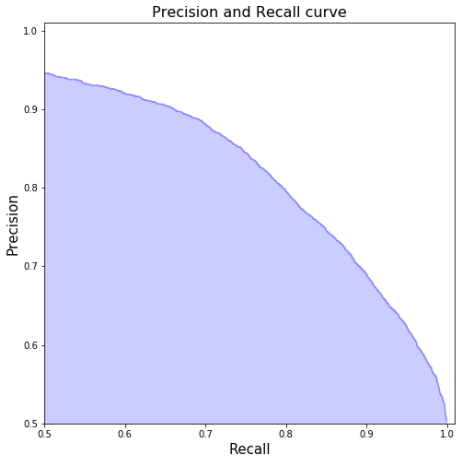

Figure 7. Curva de *Precision* / *Recall*.

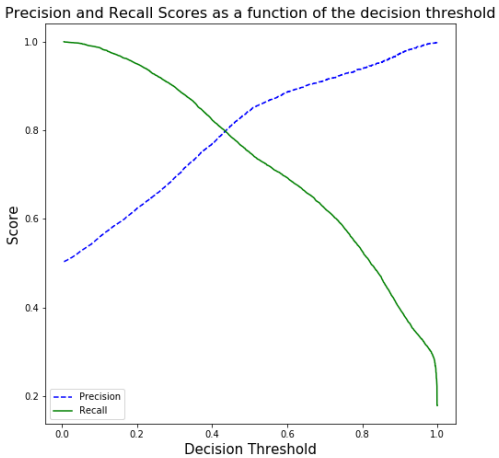

Figure 8. Valores de *Precision* e *Recall* em função do limiar de classificação.

Para o treinamento do classificador foram utilizados os parâmetros apresentados na Tabela 2.

Parâmetro	Definição
n_estimators	1000
max_depth	30
class_weight	“balanced”
random_state	42

Table 2. Parâmetros usados no algoritmo Florestas Randômicas

O algoritmo de Florestas Randômicas proporcionou uma análise sobre a importância de cada características. As Figuras 9 e 10 apresentam as 10 características mais importantes identificadas pelo algoritmo. Na Figura 10 também é incluído o desvio padrão das importâncias das características. As 10 características mais importantes são:

1. C2.3 Número de Nomes Registrados.
2. C2.11 Idade do Domínio.
3. C2.7 Número de Servidores Registrados.
4. C2.8 Número de Registradores Registrados.
5. C1.1 Tamanho do SSD.
6. C1.5 Menor número de caracteres distintos que compõe a maioria do SSD.
7. C1.9 Tamanho da maior sequência de mesmos caracteres no SUB.
8. C1.1 Tamanho do SUB.
9. C1.6 Porcentagem do SSD que os 5 caracteres mais utilizados compreendem.
10. C2.12 Menor Distância do SLD para SLD Famosos.

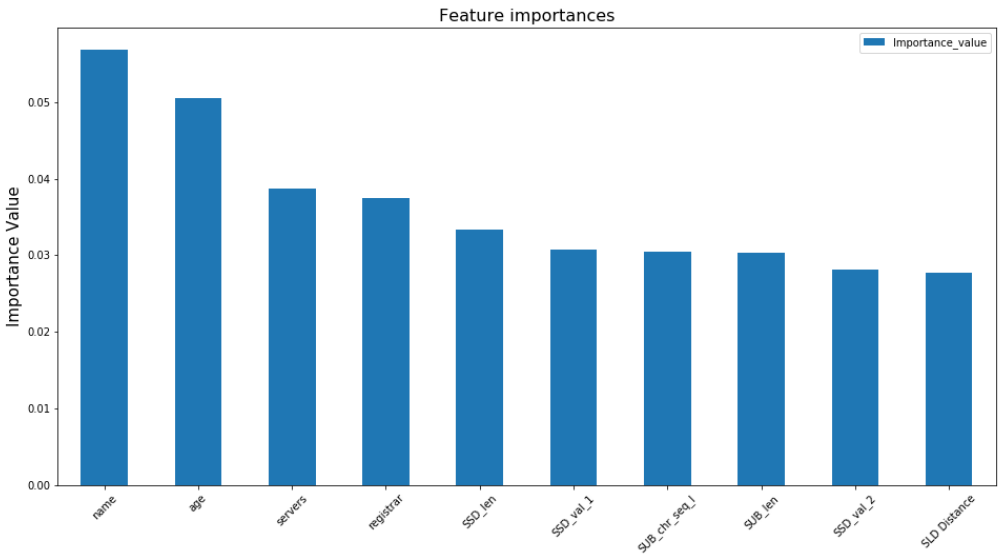

Figure 9. As dez característica mais importantes

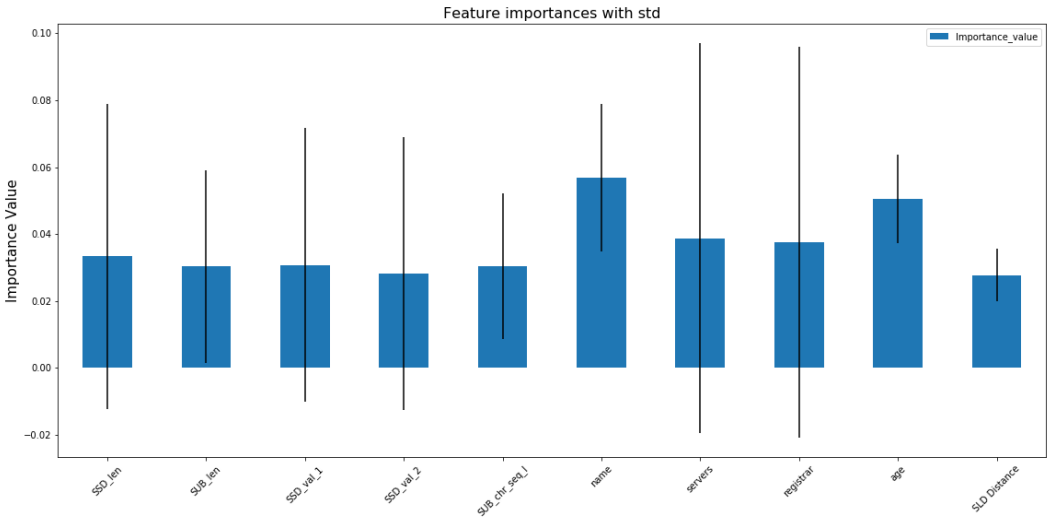

Figure 10. Desvio padrão das dez características mais importantes

3.2. Algoritmo Perceptron Multicamadas (MLP)

O segundo algoritmo testado foi o Perceptron Multicamadas (MLP). Para a definição do limiar da classificação foram analisadas as Figuras 11 e 12, uma vez que o algoritmo será utilizado juntamente com uma lista branca, desejamos um balanceamento entre métricas de *Precision* e *Recall*, estipulando assim um limiar de 0.466.

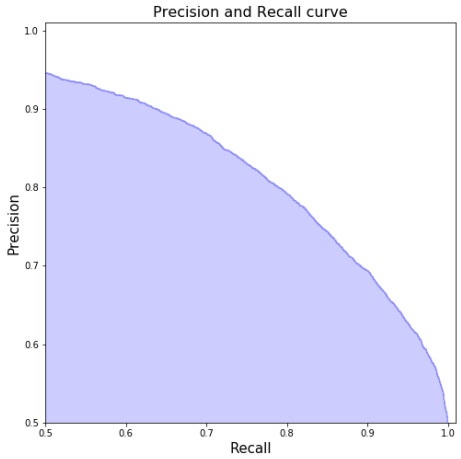

Figure 11. Curva de *Precision* / *Recall*.

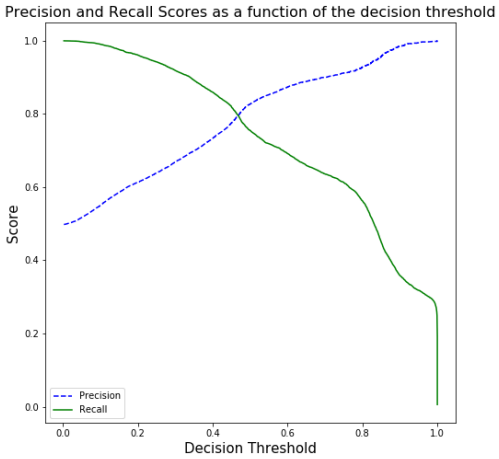

Figure 12. Valores de *Precision* e *Recall* em função do limiar de classificação.

Para o treinamento do classificador foram utilizados os parâmetros apresentados na Tabela 3.

Parâmetro	Definição
hidden_layer_sizes	$(50, 100, 50)$
random_state	42
max_iter	1000
learning_rate_init	0.001
learning_rate	"adaptive"
validation_fraction	0.1
batch_size	32
early_stopping	True
verbose	True

Table 3. Parâmetros usados no algoritmo Florestas Randômicas

As Figuras 13 e 14 apresentam as curvas de aprendizado do algoritmo. Nota-se que, por mais que a *loss* esteja diminuindo a métrica, *val_score* não está melhorando, portanto o treinamento foi interrompido automaticamente.

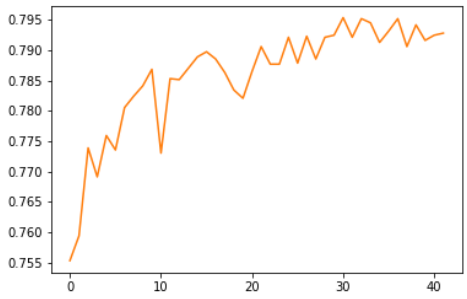

Figure 13. Curva de aprendizado do classificador utilizando a métrica *val_score*.

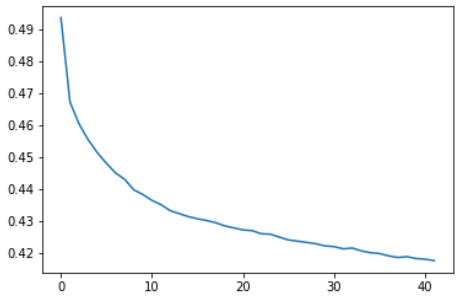

Figure 14. Curva de aprendizado do classificador utilizando a métrica *loss*.

3.3. Algoritmo K-Nearest Neighbors (KNN)

O terceiro algoritmo testado foi o K-Nearest Neighbors. Para o valor de K, principal parâmetro usado pelo algoritmo KNN, foram feitos cálculos para a taxa de erro nos dados de teste, com valores entre 1 e 10 . A Figura 15 mostra o erro aproximado para cada valor de K calculado. Com média de erro de 0.5, o melhor valor calculado foi para k=8.

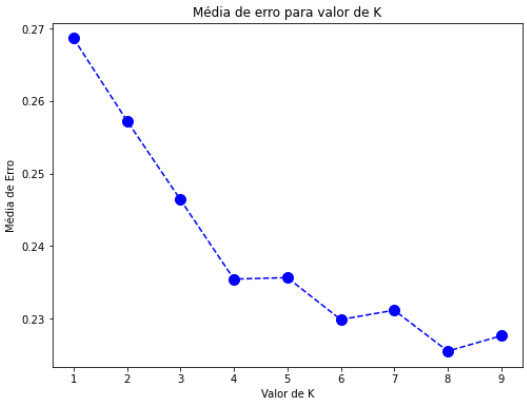

Figure 15. Valor de K pela Média de erro

Para a definição do limiar da classificação foram testados valores máximos, para que as classes malignas pudessem ser reconhecidas sem grandes perdas para identificação das classes benignas. Ao final dos testes o limiar foi definido em 0.4. As Figuras 16 e 17 representam graficamente o *Precision* e *Recall* após o ajuste do limar de classificação.

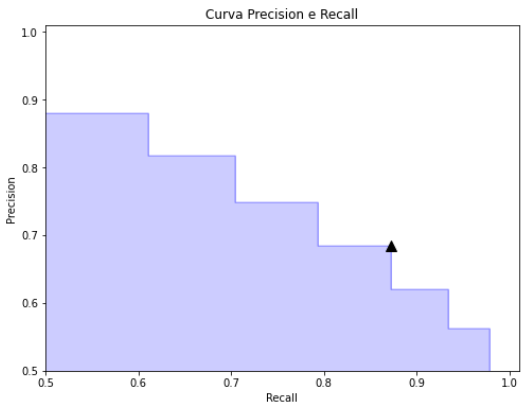

Figure 16. Curva *Precision Recall*.

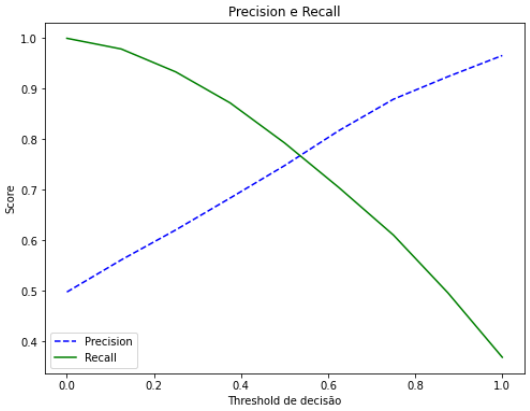

Figure 17. *Precision* e *Recall* em função do limiar de classificação.

4. Resultados

Esta seção está destinada a apresentar os resultados do algoritmos utilizados, como métricas *Precision* e *Recall*, matrizes de confusão, curva ROC, entre outros.

4.1. Validação Cruzada e Outras Métricas

O algoritmo Floresta Randômica atingiu os melhores valores para todas as métricas, quando se considera a 3a casa decimal. O algoritmo KNN teve o pior rendimento dos três testados para o modelo.

Algoritmo	Acurácia	*Precision*	*Recall*	*F1-Score*
Florestas Randômicas	0.799	0.799	0.798	0.799
Neurônio Multicamadas	0.797	0.798	0.795	0.796
K-Nearest Neighbors	0.763	0.748	0.792	0.769

Table 4. Métricas Acurácia, *Precision*, *Recall* e *F1-Score* para os algoritmos utilizados

Algoritmo	Pasta 1	Pasta 2	Pasta 3	Pasta 4	Pasta 5	Média
Florestas Randômicas	0.799	0.796	0.797	0.793	0.797	0.796
Neurônio Multicamadas	0.792	0.788	0.793	0.786	0.794	0.791
K-Nearest Neighbors	0.772	0.769	0.771	0.764	0.765	0.768

Table 5. Resultados *K-Fold Cross Validation*

4.2. Curvas ROC

A AUC *(Area Under the Curve)* para os algoritmos Florestas Randômicas e Neurônio Multicamadas mostrou que ambos tem maior capacidade em identificar classes verdadeiras em comparação ao KNN. A tabela 6 apresenta as métricas AUC de cada algoritmo, enquanto as Figuras 18, 19, 20 apresentam as curvas ROC geradas.

Algoritmo	AUC
Florestas Randômicas	0.883
Neurônio Multicamadas	0.882
K-Nearest Neighbors	0.852

Table 6. Área sob a curva (AUC) dos algoritmos utilizados

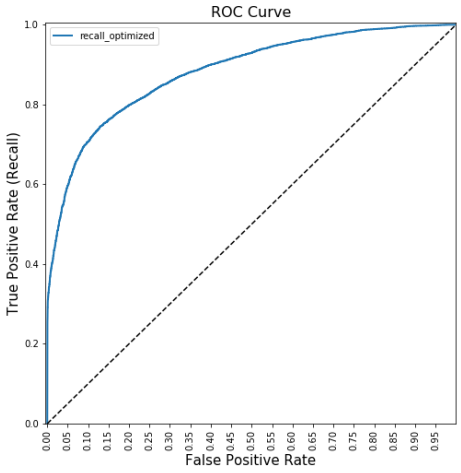

Figure 18. Curva ROC do algoritmo MLP.

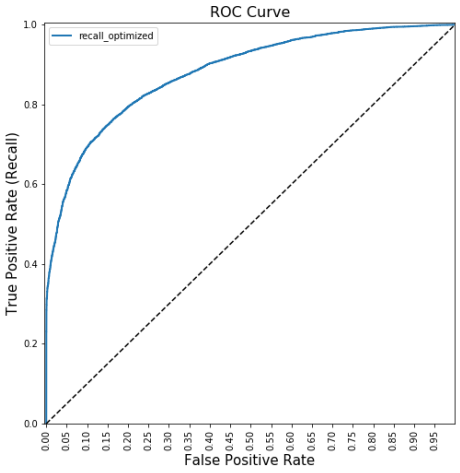

Figure 19. Curva ROC do algoritmo Floresta Randômica.

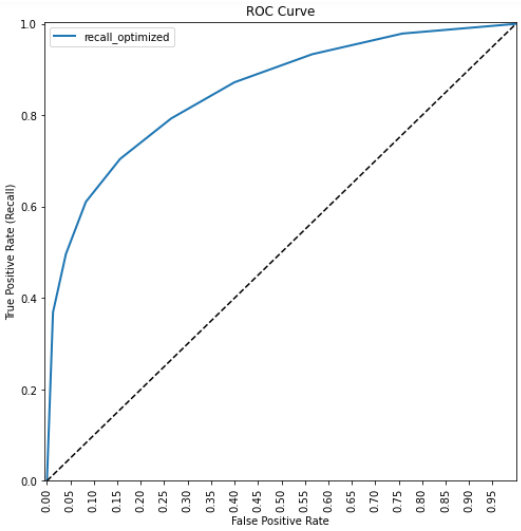

Figure 20. Curva ROC do algoritmo K-Nearest Neighbors

4.3. Matrizes de Confusão

Os resultados mostrados nas Matrizes de Confusão para os Algoritmo Floresta Negra e MLP são iguais e muito próximos para o algoritmo KNN para Verdadeiro Positivo. Eles foram buscados intencionalmente para atingir o melhor valor, pois representa a identificação correta de um domínio malicioso. Nesta condição proposta, o KNN ficou com a maior probabilidade de identificar Falsos Positivos.

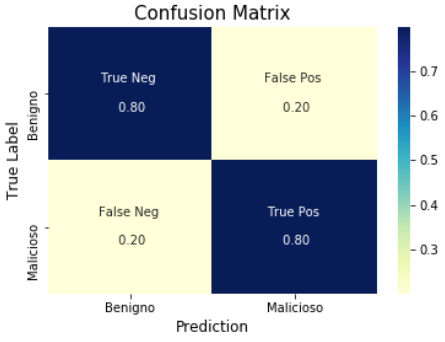

Figure 21. Matriz de Confusão do algoritmo Florestas Randomicas

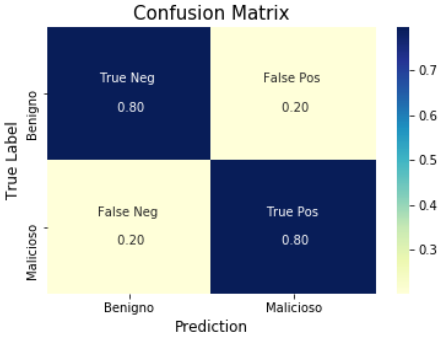

Figure 22. Matriz de Confusão

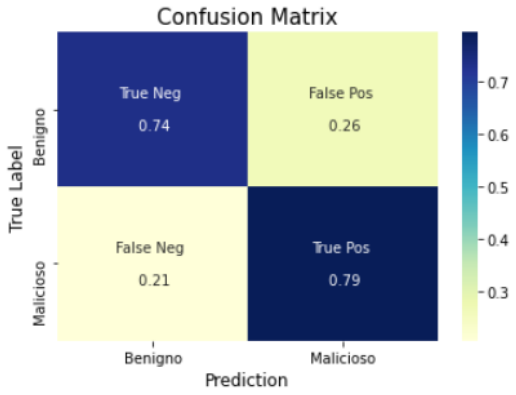

Figure 23. Matriz de Confusão do algoritmo K-Nearest Neighbors

5. Apresentação

A apresentação do trabalho foi realizada em sala.

6. Reprodutibilidade

O conjunto de dados processado, o código produzido e a documentação associada das etapas do projeto estão disponíveis em `https://github.com/AbnerBissolli/Malicious-DNS-Detection`.

7. Conclusão

Através dos classificadores foi possível construir um algoritmo para identificar se um domínio é benigno ou não. No entanto, o problema ainda está longe de ser solucionado. Neste estudo, foram testadas abordagens de classificação a partir de características lexicais e de terceiros com (*whois*m *ipwhois* e ranking *Alexa*). As possibilidades de características para a análise léxica são muitas e as trabalhadas aqui não foram suficientes para uma boa classificação. Outras abordagens podem ser utilizadas para gerar novas características, como os dados presente na seção de resposta da resposta do DNS [Mahdavifar et al. 2021]. Outra possibilidade é explorar aspectos textuais dos atributos, utilizando algoritmo como $N - grams$ e redes neurais recorrentes (RNN/LSTM), utilizando como entrada palavras presentes no nome do domínio, ou então subdivisões deste.

Nas análises o algoritmo KNN atingiu os piores índices de classificação e o Florestas Randômicas teve resultados muito próximos do Neurônio Multicamadas. Todos eles passaram por testes de limiar sobre as métricas *Precision* e *Recall* para fornecerem valores de identificação de ocorrências maliciosas próximas de 80%. O indice poderia ainda ser melhorado, mas indicaria índice pior na identificação dos domínios benignos.

Contudo, este algoritmo pode ser utilizado em conjunto com listas brancas, que permitem o acesso a sites benignos já previstos, para construir um aplicação que auxilia usuários, e pode ser utilizada como medida de prevenção antes de acessar o domínio duvidoso. Utilizando duas abordagens combinadas, a experiência do usuário será menos prejudicada com falsos alarmes, em vista da lista branca e, com taxa de *Recall* beirando os 80%, muitos domínios maliciosos serão corretamente identificados.

8. Referências

References

Amazon (2021). Top 1000 domains. `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`. Acessado em 07/12/2021.

Bezerra, M. A. et al. (2015). Uma investigação do uso de características na tetecção de urls.

Ceschin, F., Oliveira, L. S., and Grégio, A. (2019). Aprendizado de máquina para segurança: Algoritmos e aplicações. In *Minicursos do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, Dover phoenix editions, chapter Capítulo 2, pages 41–90. SBSeg.

Davison, J., Moffitt, T., and and, H. L. (2019). Webroot cybersecurity. threat report - mid-year update.

Mahdavifar, S., Maleki, N., Lashkari, A. H., Broda, M., and Razavi, A. H. (2021). Classifying malicious domains using dns traffic analysis. *The 19th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC)*.

Olivo, C. K., Santin, A., and Oliveira, L. (2010). Avaliação de características para detecção de phishing de e-mail. *Pontifícia Universidade Católica do Paraná, Curitiba–PR, Brasil*.

9. Apêndice A

9.1. Características Geradas

Table 7. Características Lexicográficas

Índice	Descrição	Tipo
C1.1	Tamanho da *string* (SSD, SUB, SLD)	*Integer*
C1.2	Porcentagem de dígitos na *string* (SSD, SUB, SLD)	*Float*
C1.3	Número de caracteres não alfa-numéricos na *string* (SSD, SUB, SLD)	*Integer*
C1.4	Entropia de Shannon da *string* (SSD, SUB, SLD)	*Float*
C1.5	Menor número de caracteres distintos que compõe a maioria da *string* (SSD, SUB, SLD)	*Integer*
C1.6	Porcentagem da *string* que os 5 caracteres mais utilizados compreendem (SSD, SUB, SLD)	*Float*
C1.7	Caractere mais utilizado na *string* (SSD, SUB, SLD)	*Char*
C1.8	Tamanho da maior sequência de dígitos na *string* (SSD, SUB, SLD)	*Integer*
C1.9	Tamanho da maior sequência de mesmos caracteres na *string* (SSD, SUB, SLD)	*Integer*
C1.10	Caractere da maior sequência de mesmos caracteres na *string* (SSD, SUB, SLD)	*Char*
C1.11	Número de palavras ruins na *string* (SSD, SUB, SLD	*Integer*
C1.12	Número de palavras alvo na *string* (SSD, SUB, SLD	*Integer*
C1.13	Número de palavras ruins, com substituição de números por letras, na *string* (SSD, SUB, SLD)	*Integer*
C1.14	Número de palavras alvo, com substituição de números por letras, na *string* (SSD, SUB, SLD)	*Integer*
C1.15	Codificação do TLD usando a função *LabelEncoder* da biblioteca *sklearn*	*Integer*
C1.16	Codificação TF-IDF do TLD usando a função *TfidfVectorizer* da biblioteca *sklearn*	*Float*
C1.17	Número de SUBS no SUB	*Integer*

Table 8. Características de Terceiros

Índice	Descrição	Tipo
C2.1	Conexão Estabelecida	*Boolean*
C2.2	Possui Inormação Privada	*Boolean*
C2.3	Número de Nomes Registrados	*Integer*
C2.4	Número de Estados Registrados	*Integer*
C2.5	Número de Países Registrados	*Integer*
C2.6	Número de Emails Registrados	*Integer*
C2.7	Número de Servidores Registrados	*Integer*
C2.8	Número de Registradores Registrados	*Integer*
C2.9	Número de Registrantes Registrados	*Integer*
C2.10	Número de Organizações Registradas	*Integer*
C2.11	Idade do Domínio	*Integer*
C2.12	Menor Distância do SLD para SLD Famosos	*Integer*
C2.13	Número de SLD Famosos no SUB	*Integer*