Universidade Federal do Rio de Janeiro

Fabiano de Moraes Domingues

Modelo de classificação baseado em aprendizagem profunda para detecção de sites falsos usados em ataques de phishing

Rio de Janeiro

2022

FABIANO DE MORAES DOMINGUES

MODELO DE CLASSIFICAÇÃO BASEADO EM APRENDIZAGEM PROFUNDA PARA DETECÇÃO DE SITES FALSOS USADOS EM ATAQUES DE PHISHING

Monografia apresentada para obtenção do título de Especialista em Ciência de Dados, no Curso de Pós-graduação Lato Sensu em Ciência de Dados da Universidade Federal do Rio de Janeiro.

Orientador: Prof. Ph.D. CARLOS TADEU PAGANI ZANINI

Rio de Janeiro

2022

FABIANO DE MORAES DOMINGUES

MODELO DE CLASSIFICAÇÃO BASEADO EM APRENDIZAGEM PROFUNDA PARA DETECÇÃO DE SITES FALSOS USADOS EM ATAQUES DE PHISHING

Monografia apresentada para obtenção do título de Especialista em Ciência de Dados, no Curso de Pós-graduação Lato Sensu em Ciência de Dados da Universidade Federal do Rio de Janeiro.

Aprovada em _____ / _____ / _____.

BANCA EXAMINADORA

Prof. Ph.D. CARLOS TADEU PAGANI ZANINI - Orientador
Universidade Federal do Rio de Janeiro

Prof. D.Sc. XXX
Universidade Federal do Rio de Janeiro

Prof. D.Sc. XXX
Universidade Federal do Rio de Janeiro

Rio de Janeiro
2022

Agradecimentos

Agradeço ao meu orientador Prof. Ph.D. Carlos Tadeu Pagani Zanini por me incentivar, apoiar e respeitar durante todo o período em que realizamos juntos a pesquisa, cujo resultado será apresentado nesta monografia.

Agradeço a todos os professores e funcionários do curso de Especialização em Ciência de Dados da Universidade Federal do Rio de Janeiro.

Lista de Figuras

Lista de Tabelas

Sumário

Resumo

O Phishing é um tipo de crime cibernético que utiliza técnicas de engenharia social para enganar usuários da internet e obter suas informações confidenciais. Normalmente, se baseia em mecanismos tecnológicos que direcionam estes usuários para páginas falsas, de fontes aparentemente confiáveis, para capturar dados como senhas e números de cartões de crédito, por exemplo. Neste trabalho, técnicas de aprendizagem profunda serão aplicadas sobre um conjunto de dados que contém características extraídas a partir de sites de phishing e legítimos, resultando em um modelo de classificação que pode ser integrado aos mecanismos de detecção e bloqueio de sites falsos.

Palavras-chave: Aprendizado de máquina, phishing, redes neurais.

Abstract

Phishing is a type of cybercrime that uses social engineering techniques to trick internet users into obtaining their confidential information. Usually, it is based on technological mechanisms that direct these users to fake pages, from apparently reliable sources, to capture data such as passwords and credit card numbers, for example. In this work, deep learning techniques will be applied on a dataset containing extracted features from phishing and legitimate websites, resulting in a classification model that can be integrated with the mechanisms for detecting and blocking fake websites.

Keywords: Machine learning, phishing, neural networks.

Capítulo 1

Introdução

As organizações modernas enfrentam um cenário de constante pressão e elevada competitividade nos negócios. A transformação digital impulsiona o uso das tecnologias de informação e dos serviços da internet na integração com os processos de negócio das organizações. Por outro lado, expõe seus sistemas de informação a uma grande variedade de ameaças e vulnerabilidades, pois a informação se tornou um ativo essencial para os negócios, e requer a proteção adequada contra diversas ameaças cibernéticas.

A ABNT ISO/IEC 27002 (2013) define uma ameaça como a causa potencial de um incidente, que pode resultar em dano para o negócio, e uma vulnerabilidade como a fragilidade de um ativo que pode ser explorada por uma ou mais ameaças. A proteção da informação contra vários tipos de ameaças pode ser obtida a partir da implementação de controles adequados, para garantir a confidencialidade, a integridade e a disponibilidade do sistema de informações.

A definição dos requisitos de proteção e a escolha dos controles de segurança adequados a serem implementados envolve a compreensão da organização sobre a motivação dos hackers maliciosos em atacar seus sistemas de informação. A NIST SP 800-12 (2017) define o hacker malicioso como um termo usado para descrever o indivíduo que usa seus conhecimento sobre computação, redes e programação para acessar sistemas ilegalmente, causar danos ou roubar informações.

Em uma abordagem com maior detalhamento sobre o conceito de hacker malicioso, a NIST SP 800-82 (2015) o define como uma ampla categoria de ameaças que podem ser divididas em classes menores, dependendo das ações específicas ou da intenção do hacker. Os grupos criminosos constituem a categoria dos hackers maliciosos que procuram atacar os sistemas com o objetivo de obterem ganhos financeiros, usando spams, phishings e spywares para cometer roubos de identidades e fraudes na internet.

O ataque de phishing se baseia no uso da engenharia social, uma técnica que depende de algum tipo de interação humana para influenciar a vítima a violar um protocolo de segurança ou incentivá-la a divulgar informações confidenciais. Na internet, o criminoso se disfarça de empresa legítima ou pessoa respeitável para induzir a vítima a acessar um site falso executar alguma ação benéfica para o hacker, como clicar em um link ou divulgar informações pessoais em um formulário falso. Este tipo de ataque também pode ser realizado através de uma solicitação fraudulenta por e-mail. (NIST SP 800-12, 2017)

O roubo ou a divulgação não autorizada de informações resulta em perda da confidencialidade,

um dos princípios básicos de segurança da informação. Segundo Stallings (2017), o objetivo da confidencialidade é preservar as restrições de autorização de acesso e divulgação de informações, incluindo meios para proteger a privacidade pessoal e informações proprietárias. A confidencialidade dos dados garante que informações privadas ou confidenciais não sejam disponibilizadas ou divulgadas a indivíduos não autorizados. A privacidade garante que os indivíduos controlem quais informações relacionadas a eles podem ser coletadas e armazenadas e por quem e para quem essas informações podem ser divulgadas.

A detecção de sites falsos usados em phishings tem sido suportada por técnicas de aprendizagem de máquina. Chiew et al. (2019) desenvolveram um novo framework de seleção das principais características usando o conjunto de dados de Tan (2018) para o sistema de detecção de sites falsos baseado em aprendizado de máquina, usando classificadores baseados em SVM, Naive Bayes, C4.5, Random Forest, JRip, e PART. Os resultados dos experimentos sugerem que o framework tem melhor desempenho com o classificador Random Forest, onde os recursos de baseline distinguem corretamente 94,6% dos sites de phishing e legítimos usando apenas 20,8% das features originais.

Em outro experimento apresentado por Chiew et al. (2019), as características de baseline (10 no total) utilizadas com o Random Forest superam o conjunto completo (48 no total) usadas pelos outros classificadores do framework. Os autores afirmaram que uma das possíveis contribuições futuras para o este trabalho seria investigar o impacto da seleção de atributos usando outros algoritmos de classificação.

Esta investigação passa obrigatoriamente pela aplicação das técnicas de aprendizado profundo usando o conjunto de dados completo de Tan (2018). De acordo com Lecun, Bengio e Hinton (2015), as redes neurais profundas permitem que modelos computacionais compostos de várias camadas de processamento aprendam representações de dados com vários níveis de abstração. Esta classe de modelos é capaz de estimar uma estrutura complexa em grandes conjuntos de dados usando o algoritmo de *back-propagation*, desenvolvido por Rumelhart, Hinton e Williams (1988).

Muitas aplicações de aprendizado profundo usam arquiteturas de rede neural *feedforward*, como os *perceptrons multicamadas* (MLP), que possuem conectividade total entre as camadas adjacentes, e aprendem pelo mapeamento de uma entrada de tamanho fixo para uma saída de tamanho fixo. Para ir de uma camada para a próxima, um conjunto de unidades calcula uma soma ponderada de suas entradas da camada anterior e passa o resultado por uma função não linear. (LECUN; BENGIO; HINTON, 2015)

O objetivo deste trabalho é aplicar um classificador baseado em uma rede neural de aprendizado profundo com uma arquitetura MLP, mapeando a totalidade dos atributos extraídos do conjunto de dados disponibilizado por Tan (2018) e explorado por Chiew et al. (2019) para uma saída que identifica o site como *phishing* ou legítimo. A definição da quantidade ótima de camadas da rede neural de aprendizado profundo será obtida a partir de experimentação. Um classificador baseado em regressão logística será utilizado como base de comparação com as arquiteturas *feedforward*.

O trabalho está organizado da seguinte forma. O capítulo 2 contém a fundamentação teórica necessária para a compreensão do trabalho desenvolvido. No capítulo 3 serão apresentadas a descrição do conjunto de dados utilizado e a arquitetura da MLP, além de alguns aspectos de sua implementação. O capítulo 4 contém os resultados da análise exploratória e dos experimentos realizados. Por fim, no capítulo 5 será apresentada a conclusão e sugestões de trabalhos futuros.

Capítulo 2

Revisão bibliográfica

Neste capítulo serão apresentados os conceitos fundamentais para a compreensão do problema que os ataques de phishing representam aos usuários da internet e da aplicação do uso de técnicas de aprendizagem profunda como solução adequada. Na Seção 2.1 serão apresentados conceitos que envolvem a engenharia social como método para realizar ataques de phishing, incluindo pharming. Na Seção 2.2 serão apresentados conceitos fundamentais sobre a arquitetura das redes neurais artificiais, com foco em modelos de aprendizado profundo a partir de MLPs com diversas camadas ocultas.

2.1 Phishing e Pharming

No contexto de segurança da informação, a engenharia social é uma técnica que depende da interação humana para influenciar um indivíduo a violar o protocolo de segurança e divulgar informações confidenciais. (NIST SP 800-12, 2017) Os ataques baseados nesta técnica são geralmente cometidos por telefone ou na Internet, sendo o primeiro meio o meio básico usado pela engenharia social. Por exemplo, um atacante pode induzir o atendente de uma empresa a acreditar que está conversando com um cliente existente, e fazer com que o funcionário divulgue informações sobre o cliente legítimo.

Na Internet, esta técnica é chamada de *phishing*, um tipo de fraude por meio da qual um atacante tenta obter dados pessoais e financeiros de um usuário, pela utilização combinada de meios técnicos e engenharia social. (CERT.br, 2012) Por exemplo, um e-mail pode ser utilizado para induzir o destinatário a realizar uma ação benéfica para o atacante, como o download de anexos que contém códigos maliciosos, clicar no link para um site falso ou divulgar informações pessoais. Um ataque de *phishing* comum é o roubo das informações de autenticação de uma vítima correspondentes a um site, que foi corrompido pelo atacante e usá-las em outro site, considerando que muitos usuários reutilizam suas senhas.

Para realizar este tipo de *phishing*, Shankar, Shetty e Nath (2019) descrevem que o atacante necessita imitar um site legítimo, construindo uma página maliciosa com o apoio de um site de *phishing*. Este site falso reune todas as informações sobre a vítima e as fornece ao invasor. Normalmente, as vítimas são incapazes de distinguir entre sites legítimos e sites de phishing, fazendo com que caiam nas armadilhas estabelecidas pelo atacante. O ciclo de vida do ataque de *phishing* foi amplamente discutido

por Damodaram (2016), Shankar, Shetty e Nath (2019), Alabdan (2020). As principais etapas do ciclo podem ser resumidas em: (ALABDAN, 2020)

1 Planejamento: identificação dos alvos e das informações buscadas; identificação e criação das ferramentas e técnicas que serão usadas no ataque, como e-mails com links maliciosos e os sites falsos para os quais esses links direcionam.

2 Phishing: ataque aos alvos identificados são executados usando os recursos criados na etapa anterior.

3 Infiltração: esta etapa pode variar de acordo com o método utilizado, mas consiste essencialmente na resposta do alvo e no acesso às informações pessoais solicitadas.

4 Coleta e exploração de dados: o atacante extrai as informações solicitadas e as utiliza para atingir os fins estabelecidos durante a fase de planejamento. Isso geralmente envolve fraude em que os invasores se passam pelas vítimas para acessar suas contas, etc. Outra ocorrência comum é a venda desses dados pessoais no mercado online.

5 Exfiltração: o atacante tenta remover o máximo possível de evidências de sua tentativa, como a exclusão de sites falsos. Também pode haver alguma análise sobre o sucesso do ataque e avaliação de ataques futuros.

O fluxo básico de um ataque de *phishing* pode ser observado Figura 2.1. Na etapa 1, o atacante cria um e-mail com o suporte de um site de phishing. Este e-mail é composto de tal forma que parece ser legítimo. Na etapa 2, o invasor envia este e-mail para a vítima. A etapa 3 indica que a vítima, incapaz de diferenciar entre e-mails legítimos e e-mails de phishing, tende a abrir a mensagem. O e-mail então a direciona para o site de phishing, e a vítima insere nele suas credenciais de login, sem saber que se trata de um site malicioso. O site de phishing fornece as credenciais de login ao invasor na etapa 4. Na última etapa, o atacante faz login no site de destino, usando os dados que obteve do site de phishing, e consegue acessar todas as informações da vítima. (SHANKAR; SHETTY; NATH, 2019)

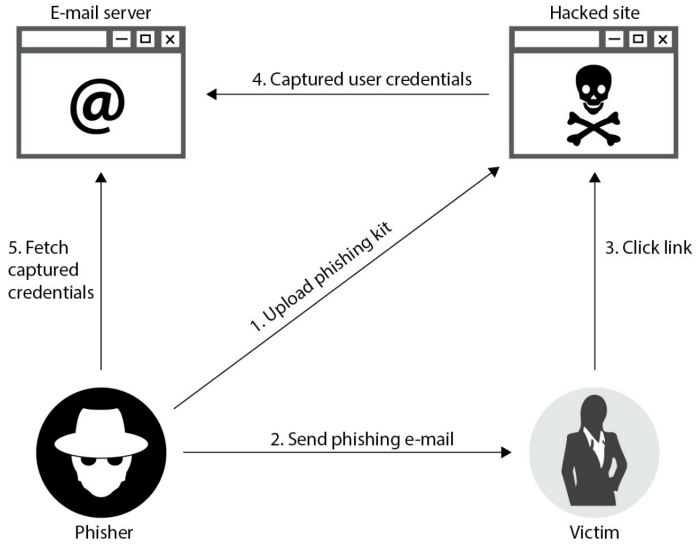


Figura 2.1: Ciclo de vida do *phishing*. (MAYMI; HARRIS, 2018)

Outros métodos incluindo o uso de sites falsos podem ser usados por um ataque de *phishing*. O *pharming* é uma variação do *phishing*, que pode vitimizar um grande número de usuários, sem direcionar o ataque a um alvo individualmente. De acordo com Damodaram (2016), o ataque de *pharming* usa os *phishings* para alterar os arquivos de *hosts* ou o sistema de nomes de domínio de tal forma que as requisições de URLs ou serviço de nomes retornam um endereço falso e as comunicações subsequentes não sabem que estão inserindo informações confidenciais em um site falso. As URLs solicitadas pelo usuário são convertidas por estes sistemas para acessar os sites, transformando os nomes dos domínios em endereços IPs. O *pharming* pode ser realizado de duas maneiras: (SHANKAR; SHETTY; NATH, 2019)

- Pelo envio de e-mail para a vítima contendo um código malicioso que modifica todos os arquivos de *hosts* do sistema operacional local. A modificação realizada pelo código malicioso faz com que o usuário seja redirecionado para o site malicioso, embora tenha inserido a URL correta no navegador.
- Nesse método, os arquivos de *hosts* do sistema operacional local do usuário não são corrompidos, mas a tabela do sistema de nomes de domínio (DNS) é modificada. Isso resulta no redirecionamento do usuário para sites maliciosos sem o seu conhecimento.

Esta técnica é chamada DNS Poisoning e o fluxo básico pode ser observado na Figura 2.2. Neste exemplo, a vítima digita o endereço www.nicebank.com no navegador. O sistema da vítima envia uma requisição para um servidor DNS envenenado, que direciona a vítima para outro site, que se parece com o site solicitado. Então, o usuário digita seu nome de usuário e senha e pode até ser apresentado a algumas páginas do site que parecem legítimas. O benefício de um ataque de pharming para o atacante é que ele pode afetar um grande número de vítimas sem a necessidade de enviar e-mails, e as vítimas geralmente acreditam nisso mais facilmente, pois estão solicitando o acesso ao site. (MAYMI; HARRIS, 2018)

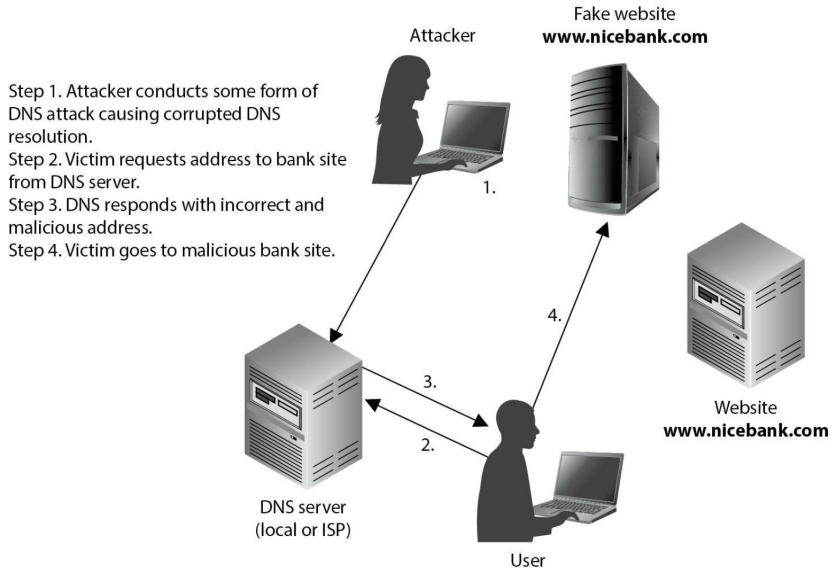


Figura 2.2: Fluxo do ataque de *pharming*. (MAYMI; HARRIS, 2018)

Os sites falsos parecem ser legítimos e são usados para coletar detalhes pessoais das vítimas quando elas tentam fazer login neles. Como os usuários comuns da Internet estão mais inclinados a

acreditar que os ataques de *phishing* são realizados principalmente através de e-mails e outros serviços de mensagens, eles tendem a ser menos preocupados com a segurança ao visitar sites, tornando-os vulneráveis a ataques baseados em sites falsos, sem o direcionamento de um e-mail malicioso. (ALABDAN, 2020)

Assim como o *pharming*, os objetivos de todas as variações de *phishings* são os mesmos. Apenas o método e a técnica utilizados para obter as informações variam de um tipo para outro (SHANKAR; SHETTY; NATH, 2019). Não existe um procedimento de defesa que possa impedir todo e qualquer ataque de *phishing*. No entanto, diferentes abordagens aplicadas em cada etapa do ciclo de vida do ataque podem mitigar uma tentativa de *phishing*. A tecnologia aplicada corretamente pode reduzir significativamente o risco de roubo de identidade (DAMODARAM, 2016).

2.2 Redes Neurais Profundas

As redes neurais artificiais simulam o funcionamento dos neurônios biológicos e permitem o aprendizado de máquina para resolução de problemas complexos e tomada de decisões inteligentes. Uma rede biológica permite que os seres humanos experimentem e interajam com o mundo ao seu redor. A unidade fundamental do cérebro humano é o neurônio, cuja função básica é receber informações de outros neurônios, processá-las e enviar os resultados para outras células.

Uma breve descrição do funcionamento do neurônio biológico foi apresentada por Buduma e Locascio (2017). O neurônio recebe suas entradas ao longo de estruturas chamadas dendritos. Cada uma dessas conexões de entrada é fortalecida ou enfraquecida dinamicamente com base na frequência com que é usada, e é a força de cada conexão que determina a contribuição da entrada para a saída do neurônio. Depois de ponderadas pela força de suas respectivas conexões, as entradas são somadas no corpo da célula. Essa soma é então transformada em um novo sinal que é propagado ao longo do axônio da célula e enviado para outros neurônios. A estrutura e o processo descritos podem ser observados na Figura 2.3.

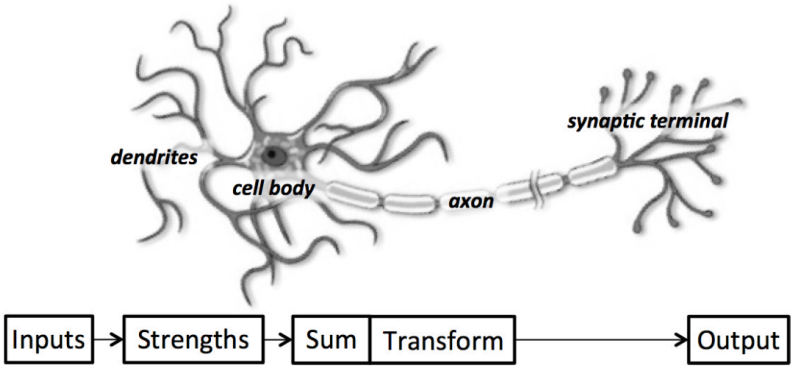


Figura 2.3: Descrição funcional da estrutura de um neurônio biológico. (BUDUMA; LOCASCIO, 2017)

As primeiras simulações computacionais de redes neurais artificiais passaram a ser desenvolvidas a partir da década de 1950, devido ao avanço dos computadores. Alguns anos antes, o modelo de uma rede neural simples baseada em circuitos elétricos foi apresentado por Mcculloch e Pitts (1943), mas sem um método de treinamento definido. O aprendizado humano foi explicado por Hebb (1949) como o fortalecimento dos caminhos neurais cada vez que são usados. Esta rede neural simples foi descrita

por Rosenblatt (1957) como um *perceptron*, e implementada como uma máquina construída por ele em 1958. Rumelhart, Hinton e Williams (1988) desenvolveram um algoritmo de treinamento chamado *backpropagation*, que armazena em memória os valores das derivadas com respeito aos pesos nas camadas superiores da rede neural para utilização no cálculo das derivadas das camadas inferiores de forma iterativa e eficiente.

Uma representação computacional de um neurônio artificial foi descrita por Heaton (2015) e pode ser observada na Figura 2.4. O neurônio artificial recebe entrada de uma ou mais fontes que podem ser outros neurônios ou dados de entrada alimentados na rede. As entradas podem ser valores contínuos (quando a variável associada é contínua) ou discretos (quando a variável associada é discreta). Variáveis discretas muitas vezes representam categorias nominais ou indicam ausência ou presença de determinada características e são sempre codificadas na forma de valores numéricos. Um neurônio artificial combina linearmente cada uma dessas entradas por um conjunto de pesos. Em seguida, o neurônio passa essa combinação linear por uma função de ativação não linear, geralmente não decrescente. Na Figura 2.4 é possível observar o processo.

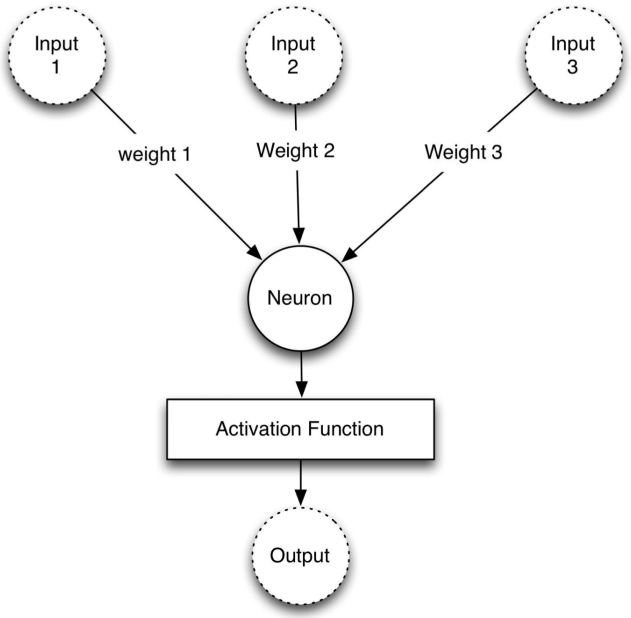


Figura 2.4: Um neurônio artificial. (HEATON, 2015)

A Equação 2.1 resume a saída calculada de um neurônio. As variáveis $\mathbf{x} = x_1, ..., x_n$ e $\mathbf{w} = w_1, ..., w_n$ representam as entradas e os pesos do neurônio. O valor n corresponde ao número de pesos e entradas. É preciso ter sempre o mesmo número de pesos e entradas. Cada peso é multiplicado por sua respectiva entrada. Portanto, o neurônio artificial deve receber as entradas, $x_1, ..., x_n$, cada uma das quais sendo multiplicada por um peso específico, $w_1, ..., w_n$. A soma desses produtos alimenta como entrada uma função de ativação, denotada aqui pela letra grega ϕ (phi). Em muitos casos, pode ser incluído um viés b, que é uma constante somada a $\sum w_i x_i$ para compor a entrada de ϕ. Este processo resulta em uma saída única do neurônio, que pode ser transmitida a outros neurônios.

$$f(\mathbf{x}, \mathbf{w}) = \phi\left(\sum_{i=1}^{n} w_i x_i\right) \tag{2.1}$$

A Equação 2.2 expressa um neurônio com pesos e viés. Vale ressaltar que as formulações em (2.1) e (2.2) são equivalentes, uma vez que é possível considerar entradas $x = x_1, ..., x_{n+1}$ e pesos $w = w_1, ..., w_{n+1}$ na Equação 2.1, onde $x_{n+1} = 1$ e $w_{n+1} = 1$.

$$f(\mathbf{x}, \mathbf{w}, b) = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.2}$$

Neurônios individuais não são expressivos o suficiente para resolver problemas de aprendizagem complexos. Para lidar com tarefas mais complicadas, Buduma e Locascio (2017) afirmam que é necessário conectar vários neurônios em uma organização estruturada por camadas em sequência, formando uma rede neural artificial. De acordo com Heaton (2015), os neurônios que formam uma camada compartilham a mesma função de ativação, mas possuem, cada um, o próprio conjunto de pesos. Diferentes camadas podem ter diferentes funções de ativação. Por fim, cada camada, exceto a última, está totalmente conectada à próxima camada, ou seja, cada neurônio em uma camada está conectado com os neurônios da camada seguinte.

Na Figura 2.5, é possível observar uma rede neural multicamadas totalmente conectada. A rede possui uma camada de entrada e saída, e duas camadas ocultas. O número de camadas ocultas determina o nome da arquitetura de rede. Tanto a primeira camada oculta quanto a segunda contém dois neurônios. Os neurônios N1 e N2 fazem parte da camada #1 enquanto que #2 possui N3 e N4. Não é necessário que as camadas possuam a mesma quantidade de neurônios. As setas apontando para baixo ou sempre para frente considerando o sentido da entrada para a saída indica que esta é uma rede *feedforward*.

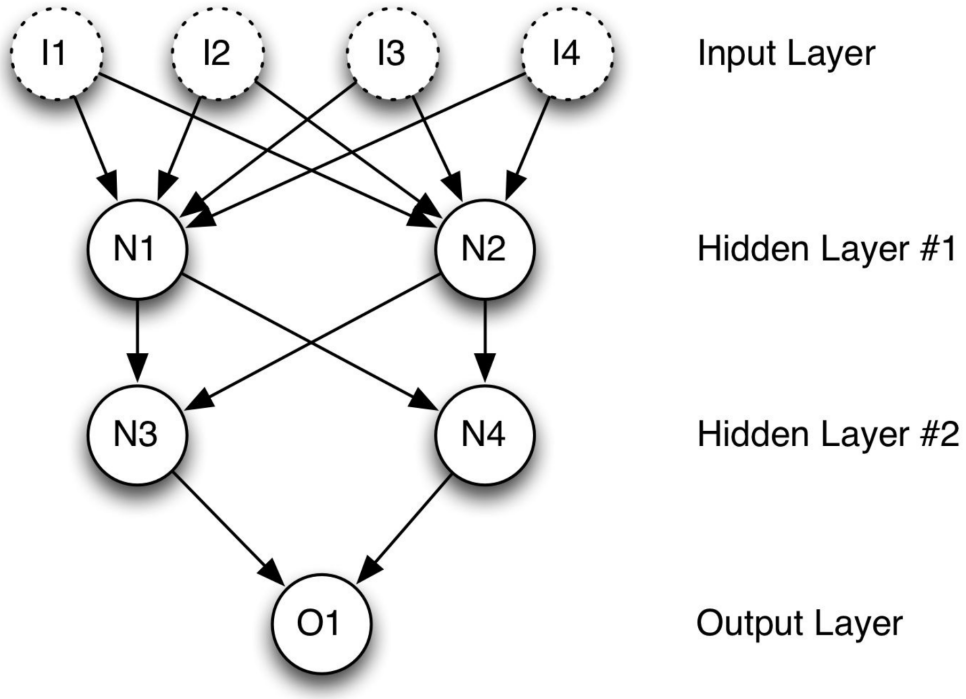


Figura 2.5: Rede neural artificial totalmente conectada. (HEATON, 2015)

Na Figura 2.6 é possível observar o fluxo dos pesos pelos neurônios em uma rede neural *feedforward* simples, com uma camada oculta e três neurônios por camada, seguindo o exemplo descrito por Buduma e Locascio (2017). Neste exemplo, o fluxo segue de baixo para cima. A primeira camada recebe os dados de entrada, enquanto que a última camada calcula a resposta final. As camadas intermediárias são também denominadas camadas ocultas, e $w_{i,j}^{(k)}$ é o peso da conexão entre o i-ésimo neurônio na k-ésima camada com o j-ésimo neurônio na $(k+1)$-ésima camada. Esses pesos constituem um vetor de parâmetros θ, e a capacidade das redes neurais em resolver problemas depende de encontrar os valores ideais de θ.

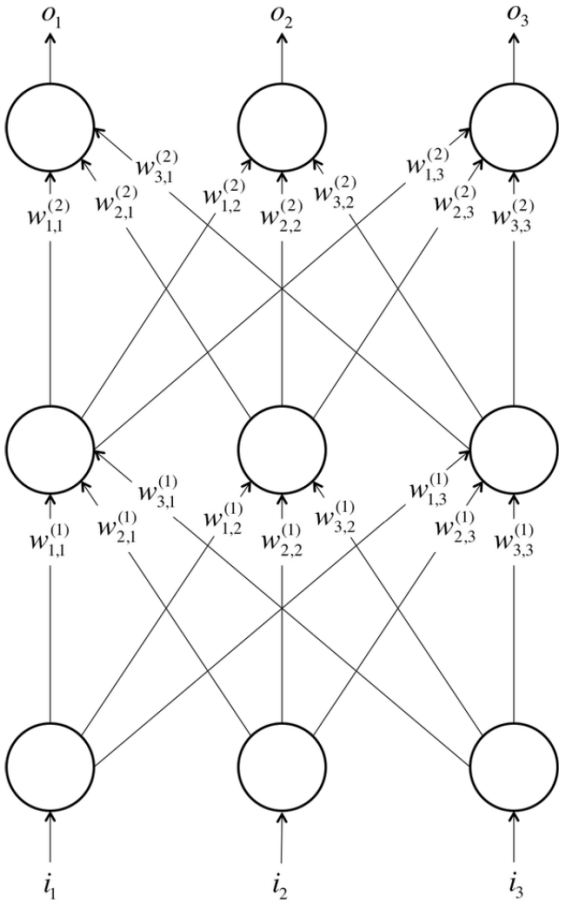


Figura 2.6: Fluxo de pesos em uma rede *feedforward* totalmente conectada e com uma camada oculta. (BUDUMA; LOCASCIO, 2017)

Este tipo de rede é chamado *Multilayer Perceptron*. (BAUM, 1988) As MLPs são redes *feedforward* dispostas em pelo menos três camadas (entrada, oculta e saída) totalmente conectadas, e usam *backpropagation* como a técnica de aprendizado. Os neurônios de entrada são os inputs, ou covariáveis, da base de dados. Os neurônios da(s) camada(s) oculta(s) e de saída possuem funções de ativação não lineares, além de pesos. O uso de funções de ativação não lineares permite que as MLPs consigam distinguir dados que não são linearmente separáveis, pois a fronteira de classificação resultante é uma função não linear. Este aspecto diferencia as MLPs de um *Perceptron* Linear, e de modelos como o de regressão logística, para os quais a fronteira de classificação é um hiperplano.

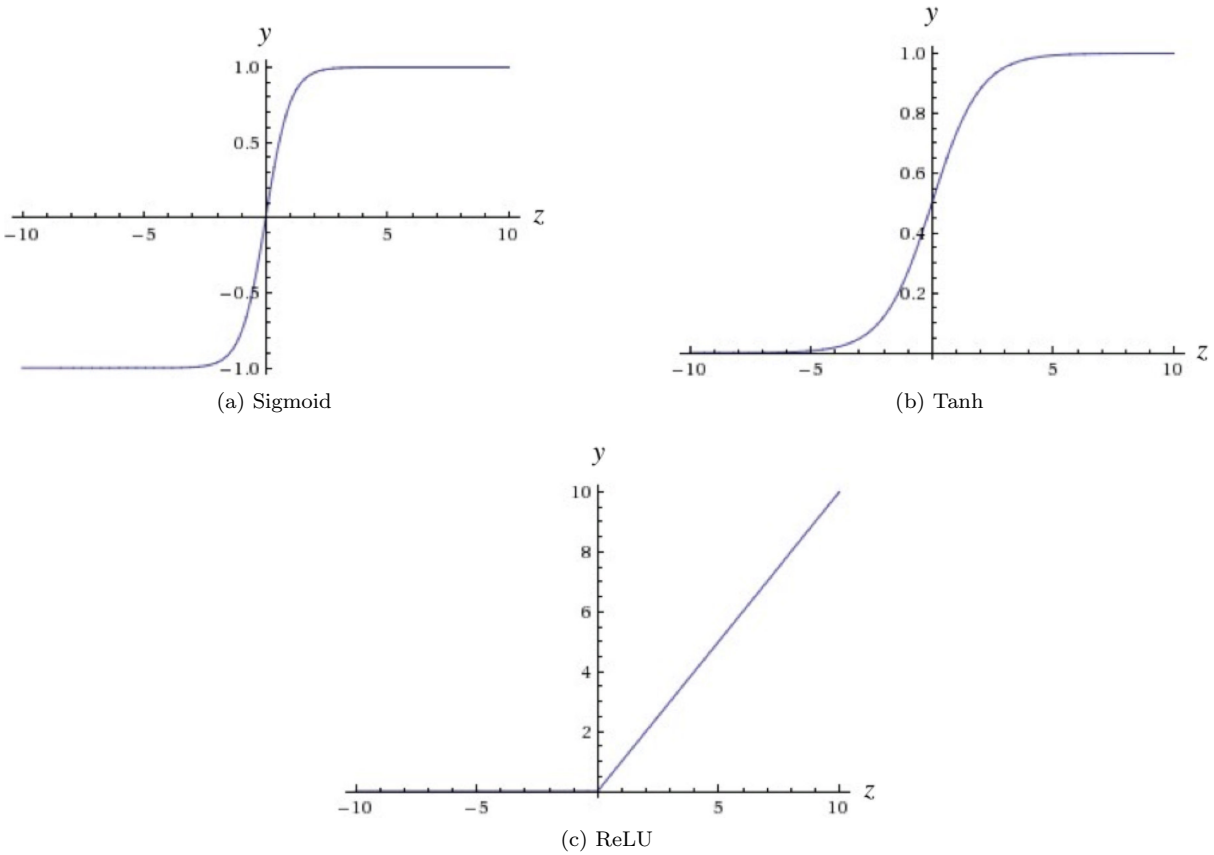
Existem três tipos principais de neurônios que são usados na prática e que introduzem não linearidades em seus cálculos. (BUDUMA; LOCASCIO, 2017) O primeiro deles usa a função de ativação

Sigmóide, baseada na Equação 2.2. Intuitivamente, quando z é muito pequeno, a saída de um neurônio é muito próxima de 0, ou próxima de 1, quando z é muito grande. Entre os extremos, o neurônio assume a forma de S. A função Tanh é baseada na Equação 2.3 e usa um tipo semelhante de não linearidade em forma de S, mas em vez de variar de 0 a 1, a saída dos neurônios varia de -1 a 1. Um tipo diferente de não linearidade é usado pelo neurônio com a função de ativação ReLU, que significa Unidade Linear Restrita, e se baseia na Equação 2.4. Na Figura 2.7, é possível verificar os gráficos das funções de ativação Sigmóide (a), Tanh (b) e ReLU (c).

$$f(z) = \frac{1}{1 + \epsilon^{-z}} \tag{2.3}$$

$$f(z) = tanh(z) \tag{2.4}$$

$$f(z) = max(0, z) \tag{2.5}$$



(a) Sigmoid

(b) Tanh

(c) ReLU

Figura 2.7: Principais funções de ativação não lineares.

Os avanços em unidades de processamento gráfico de alta velocidade (GPU) permitiram que os programadores treinassem redes neurais artificiais usando *backpropagation* com muitas camadas. Estes avanços permitiram uma evolução na precisão e no desempenho de modelos de redes neurais *feedforward* profundas (com muitas camadas) capazes de aprender padrões mais complexos (HEATON, 2015).

Capítulo 3

Aplicação ao problema de detecção de sites maliciosos

Neste capítulo serão apresentados os aspectos relacionados com a aplicação de um modelo de aprendizado profundo na detecção de sites maliciosos usados em ataques de phishing. Na Seção 3.1 será apresentada uma breve descrição de cada atributo do conjunto de dados usado nos experimentos. Na Seção 3.2 será apresentada a arquitetura da rede neural multilayer perceptron a ser aplicada, que envolve tratar das camadas de entreda, ocultas e de saída, além do número de neurônios de cada uma. Por fim, serão abordados alguns aspectos da implementação desta arquitetura, como o uso de algumas bibliotecas e o código da construção do modelo.

3.1 Conjunto de dados

O conjunto de dados utilizado neste trabalho foi disponibilizado por Tan (2018), para que pesquisadores e especialistas na área de antiphishing possam realizar a análise de aspectos característicos do phishing, em experimentos de prova de conceito ou benchmarking de modelos para classificação de phishing. O conjunto de dados contém 48 *features* (inputs ou covariáveis) extraídas de 5.000 páginas da Internet usadas em ataques de *phishing* e 5.000 páginas legítimas, obtidas nos períodos entre janeiro e maio de 2015, e entre maio e junho de 2017. As features foram extraídas de URLs de páginas Web e códigos HTML. A descrição de cada feature apresentada por Chiew et al. (2019) pode ser observada a seguir:

- NumDots → Conta o número de pontos na URL da página.
- SubdomainLevel → Conta o nível de subdomínio na URL da página.
- PathLevel → Conta a profundidade do caminho na URL da página.
- UrlLength → Conta o total de caracteres na URL da página.
- NumDash → Conta o número de "-" na URL da página.
- NumDashInHostname → Conta o número de "-" na parte do nome do host na URL da página.

- AtSymbol → Verifica se o símbolo “@” existe na URL da página.
- TildeSymbol → Verifica se o símbolo “~” existe na URL da página.
- NumUnderscore → Conta o número de “_” na URL da página.
- NumPercent → Conta o número de “%” na URL da página.
- NumQueryComponents → Conta o número de partes de consulta na URL da página.
- NumAmpersand → Conta o número de “&” na URL da página.
- NumHash → Conta o número de “#” na URL da página.
- NumNumericChars → Conta o número de caracteres numéricos na URL da página.
- NoHttps → Verifica se não existe HTTPS na URL da página.
- RandomString → Verifica se existem strings aleatórias na URL da página.
- IpAddress → Verifica se o endereço IP é usado na parte do nome do host da URL da página.
- DomainInSubdomains → Verifica se o TLD ou ccTLD é usado como parte do subdomínio na URL da página.
- DomainInPaths → Verifica se TLD ou ccTLD é usado no caminho do URL da página.
- HttpsInHostname → Verifica se o HTTPS está ofuscado na parte do nome do host na URL da página.
- HostnameLength → Conta o total de caracteres na parte do nome do host na URL da página.
- PathLength → Conta o total de caracteres no caminho da URL da página.
- QueryLength → Conta o total de caracteres na parte de consulta na URL da página.
- DoubleSlashInPath → Verifica se existe “//” no caminho da URL da página.
- NumSensitiveWords → Conta o número de palavras confidenciais (ou seja, ”seguro”, ”conta”, ”webscr”, ”login”, ”ebayisapi”, ”login”, ”banco”, ”confirmar”) na URL da página.
- EmbeddedBrandName → Verifica se o nome da marca aparece nos subdomínios e o caminho da URL da página. O nome da marca aqui é assumido como o nome de domínio mais frequente no conteúdo HTML da página.
- PctExtHyperlinks → Conta a porcentagem de hiperlinks externos no código HTML da página.
- PctExtResourceUrls → Conta a porcentagem de URLs de recursos externos no código HTML da página.
- ExtFavicon → Verifica se o favicon é carregado a partir de um nome de domínio diferente do nome de domínio na URL da página.

- InsecureForms → Verifica se o atributo de ação do formulário contém uma URL sem protocolo HTTPS.
- RelativeFormAction → Verifica se o atributo de ação do formulário contém uma URL relativa.
- ExtFormAction → Verifica se o atributo de ação do formulário contém uma URL de um domínio externo.
- AbnormalFormAction → Verifica se o atributo de ação do formulário contém um "#", "about:blank", uma string vazia ou "javascript:true".
- PctNullSelfRedirectHyperlinks → Conta a porcentagem de campos de hiperlinks contendo valor vazio, valor de autorredirecionamento como "#", o URL da página atual ou algum valor anormal como "arquivo://E:/".
- FrequentDomainNameMismatch → Verifica se o nome de domínio mais frequente no código HTML não corresponde ao nome de domínio na URL da página.
- FakeLinkInStatusBar → Verifica se o código HTML contém o comando JavaScript onMouseOver para exibir uma URL falsa na barra de status.
- RightClickDisabled → Verifica se o código HTML contém o comando JavaScript para desativar a função de clique com o botão direito.
- PopUpWindow → Verifica se o código HTML contém o comando JavaScript para iniciar pop-ups.
- SubmitInfoToEmail → Verifica se o código HTML contém a função HTML "mailto".
- IframeOrFrame → Verifica se iframe ou frame é usado no código HTML.
- MissingTitle → Verifica se a tag de título está vazia no código HTML.
- ImagesOnlyInForm → Verifica se o escopo do formulário no código HTML não contém nenhum texto, mas apenas imagens.
- SubdomainLevelRT → Conta o número de pontos na parte do nome do host do URL da página.
- UrlLengthRT → Conta o total de caracteres na URL da página.
- PctExtResourceUrlsRT → Conta a porcentagem de URLs de recursos externos no código HTML da página.
- AbnormalExtFormActionR → Verifica se o atributo de ação do formulário contém um domínio estrangeiro, "about:blank" ou uma string vazia.
- ExtMetaScriptLinkRT → Conta a porcentagem de meta, script e tags de link contendo URL externa nos atributos.
- PctExtNullSelfRedirectHyperlinksRT → Conta a porcentagem de hiperlinks no código HTML que usa nomes de domínio diferentes, começa com "#" ou usa "JavaScript ::void(0)".

O conjunto de dados possui um identificador numérico representado pelo atributo id, e outro que representa o rótulo, nomeado como CLASS_LABEL, que armazena valores binários, e identifica um site falso ou phishing como 1, e legítimo como 0. A descrição dos dados apresentada por Chiew et al. (2019) também categoriza os tipos de dados de cada *feature*, como: binário, categórico, contínuo ou discreto. Este último se refere a variáveis numéricas com mais de dois valores. As relações entre as *features* e os tipos de dados que armazenam pode ser observada na Tabela 3.1.

Tabela 3.1: Categorização das features por tipo de dado. (CHIEW et al., 2019)

Tipo	**Features**
Binário	AtSymbol, TildeSymbol, NoHttps, RandomString, IpAddress, DomainInSubdomains, DomainInPaths, HttpsInHostname, DoubleSlashInPath, EmbeddedBrandName, ExtFavicon, InsecureForms, RelativeFormAction, ExtFormAction, FrequentDomainNameMismatch, FakeLinkInStatusBar, RightClickDisabled, PopUpWindow, SubmitInfoToEmail, IframeOrFrame, MissingTitle e ImagesOnlyInForm.
Categórico	AbnormalFormAction, SubdomainLevelRT, UrlLengthRT, PctExtResourceUrlsRT, AbnormalExtFormActionR, ExtMetaScriptLinkRT e PctExtNullSelfRedirectHyperlinksRT
Contínuo	PctExtHyperlinks, PctExtResourceUrls e PctNullSelfRedirectHyperlinks.
Discreto	NumDots, SubdomainLevel, PathLevel, UrlLength, NumDash, NumDashInHostname NumUnderscore, NumPercent, NumQueryComponents, NumAmpersand, NumHash, NumNumericChars, HostnameLength, PathLength, QueryLength e NumSensitiveWords.

3.2 Arquitetura da MLP

O conjunto de dados obtido em Tan (2018) e descrito na Seção 3.1 será analisado por quatro redes neurais artificiais do tipo *Multilayer Perceptron*. A principal diferença entre elas é a quantidade de camadas ocultas, que varia de 1 a 4. Por este motivo, as MLPs podem ter suas arquiteturas abstraídas na estrutura da Figura 3.1. São 48 neurônios de entrada que correspondem às *features* do conjunto de dados. As camadas ocultas possuem 128 neurônios e função de ativação Relu. A camada e saída determina se o site é benigno, retornando valor 0, ou falso, usando em ataques de phishing, e retornando valor 1. A função de ativação na camada de saída é a Sigmoid.

A implementação da MLP utiliza como base o Keras[1], uma API de aprendizado profundo escrita em Python, e executada na plataforma de aprendizado de máquina TensorFlow[2]. A arquitetura de cada MLP é implementada no Keras como modelos, que são construídos em camadas. Em cada camada, deve ser definida a quantidade de neurônios e a função de ativação. No Keras, as camadas de Dropout sorteiam os neurônios da camada que serão excluídos de cada etapa do treinamento, o que ajuda a evitar *overfitting*. O percentual esperado de unidades de entrada a serem descartadas para cada camada do modelo é determinado por uma probabilidade p. Os demais neurônios são aumentados por um fator multiplicativo $\frac{1}{1-p)}$. Os valores de p aplicados correspondem aos propostos por Srivastava et al. (2014), sendo $p = 0.2$ para a camada de entrada e $p = 0.5$ para as ocultas.

[1]https://keras.io/
[2]https://www.tensorflow.org/

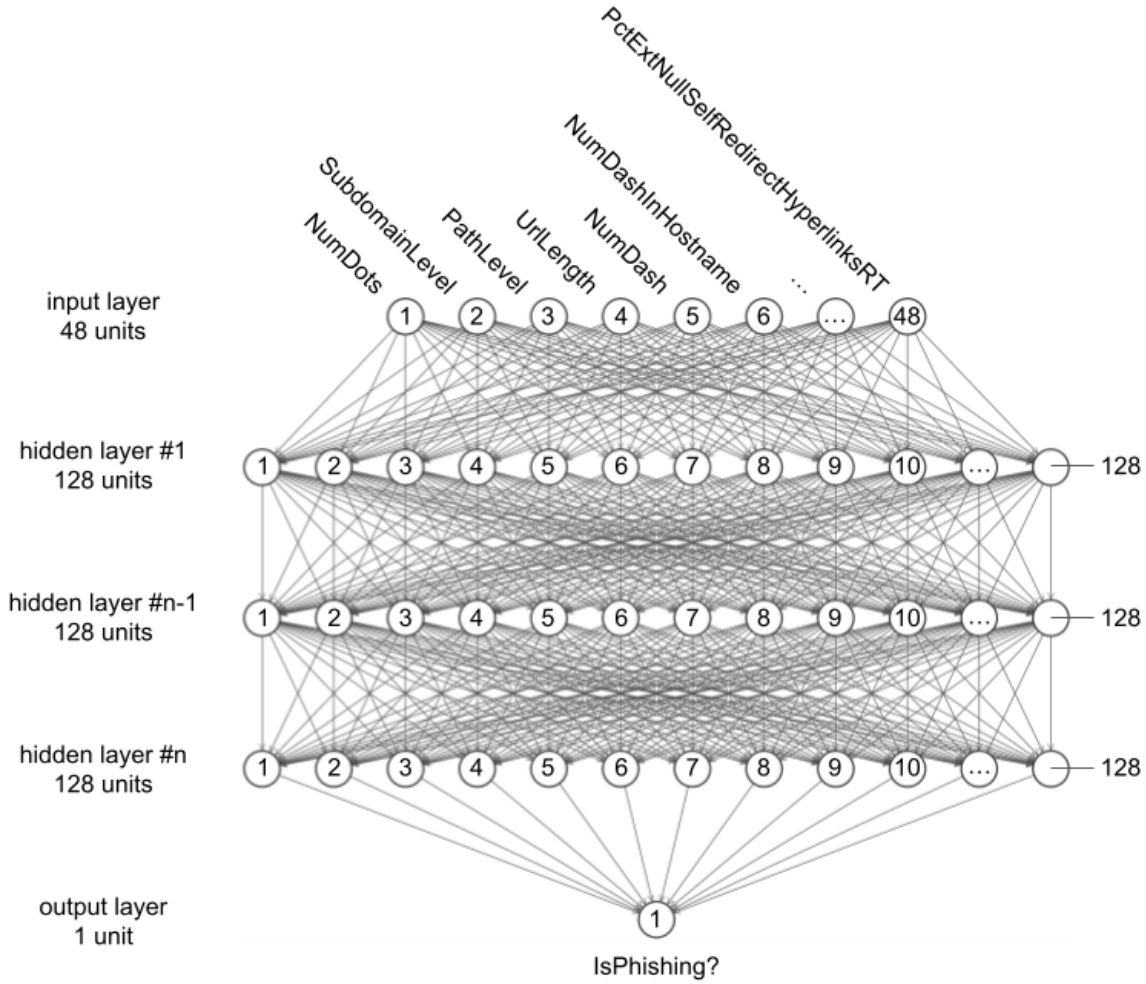


Figura 3.1: Arquitetura básica das MLPs usadas nos experimentos.

Na compilação do modelo, devem ser informados o otimizador, a função de perda e as métricas para medir seu desempenho nos subconjuntos de dados de teste e avaliação. O otimizador utilizado nos modelos é o Adam, um método de gradiente descendente estocástico modificado que se baseia na estimativa adaptativa de momentos de primeira e segunda ordem, desenvolvido por Kingma e Ba (2015), invariante ao reescalonamento diagonal de gradientes, e adequado para problemas que possuem grandes volumes de dados. Nos experimentos iniciais realizados durante o desenvolvimento das redes, o método Adam apresentou os melhores resultados e por isso foi aplicado na estimação dos parâmetros dos modelos.

O código usado para a construção de cada modelo pode ser observado no Código 3.1. A classe *Sequential* agrupa uma pilha linear de camadas em um modelo, e fornece recursos de treinamento e inferência. As instâncias das camadas são adicionadas no topo da pilha com o método *add()*. A classe *Dropout* é usada para descartar aleatoriamente unidades de entrada sobre cada camada, de acordo com o percentual informado. Uma estrutura de repetição *for* itera sobre a quantidade de camadas ocultas, definida em *hidden_layers*, onde são construídas e adicionadas ao modelo. O otimizador Adam é instanciado utilizado no método *compile()* do modelo, além da função de perda e da métrica.

```
model=Sequential()
for hl in range(hidden_layers):
    rate=0.5
    if hl==0:
        rate=0.2
    model.add(Dropout(rate))
    model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
```

Código 3.1: Código de implementação e compilação do modelo usado nos experimentos.

Capítulo 4

Experimentos realizados

Neste Capítulo serão apresentados os resultados da execução dos modelos na detecção de sites falsos usados em ataques de phishing. Na Seção 4.1, será detalhada a análise exploratória nos dados, que envolve observar a relação entre cada atributo e a classe, além da relação entre eles. Na Seção 4.2 serão apresentados os resultados dos experimentos nas fases de treinamento e avaliação dos modelos, e o mecanismo de escolha da MLP com melhor desempenho. Na Seção 4.3, o resultado dos experimentos na fase de teste, com a execução do MLP de melhor desempenho na fase anterior e a RL, será apresentado, permitindo uma compoaração entre as duas abordagens na aplicação da detecção de sites falsos.

4.1 Análise dos Dados

A análise da relação entre as *features* e a classe tem como objetivo identificar os atributos que influenciam no fato de um site ser benigno ou phishing. As relações entre os atributos binários e categóricos identificados na Tabela 3.1, e a classe, podem ser analisadas nos gráficos da Figura 4.1. O primeiro gráfico a ser observado é o do atributo *HttpsInHostname*. Seu gráfico indica que todos os valores deste atributo no conjunto de dados possuem valor zero. De acordo com a descrição do atributo apresentada na Seção 3.1, em nenhuma amostra foi verificado o HTTPS ofuscado na parte do nome do *host* do URL da página.

Além do atributo ser irrelevante para o conjunto de dados, seu valor será exatamente igual ao intercepto, e isso pode gerar problemas na estimação dos parâmetros. Pelos motivos apresentados, o atributo *HttpsInHostname* é removido do conjunto de dados. Esta decisão reflete na arquitetura da rede neural apresentada na Figura 3.1. Como resultado desta modificação, passa a ser 47 o número de neurônios na camada de entrada de cada um dos modelos usados nos experimentos.

A observação dos gráficos da Figura 4.1 permite considerar que há alguns atributos que apresentam maior influência na classe do que outras. Os atributos que se destacam são: InsecureForms, FrequentDomainNameMismatch, SubmitInfoToEmail, IframeOrFrame, UrlLengthRT, ExtMetaScriptLinkRT e PctExtNullSelfRedirectHyperlinksRT. Em menor proporção, é possível observar que os atributos RandomString, DomainInPaths, RelativeFormAction, ExtFormAction, AbnormalFormAction, PctExtResourceUrlsRT e AbnormalExtFormActionR também apresentam alguma influência sobre a classe.

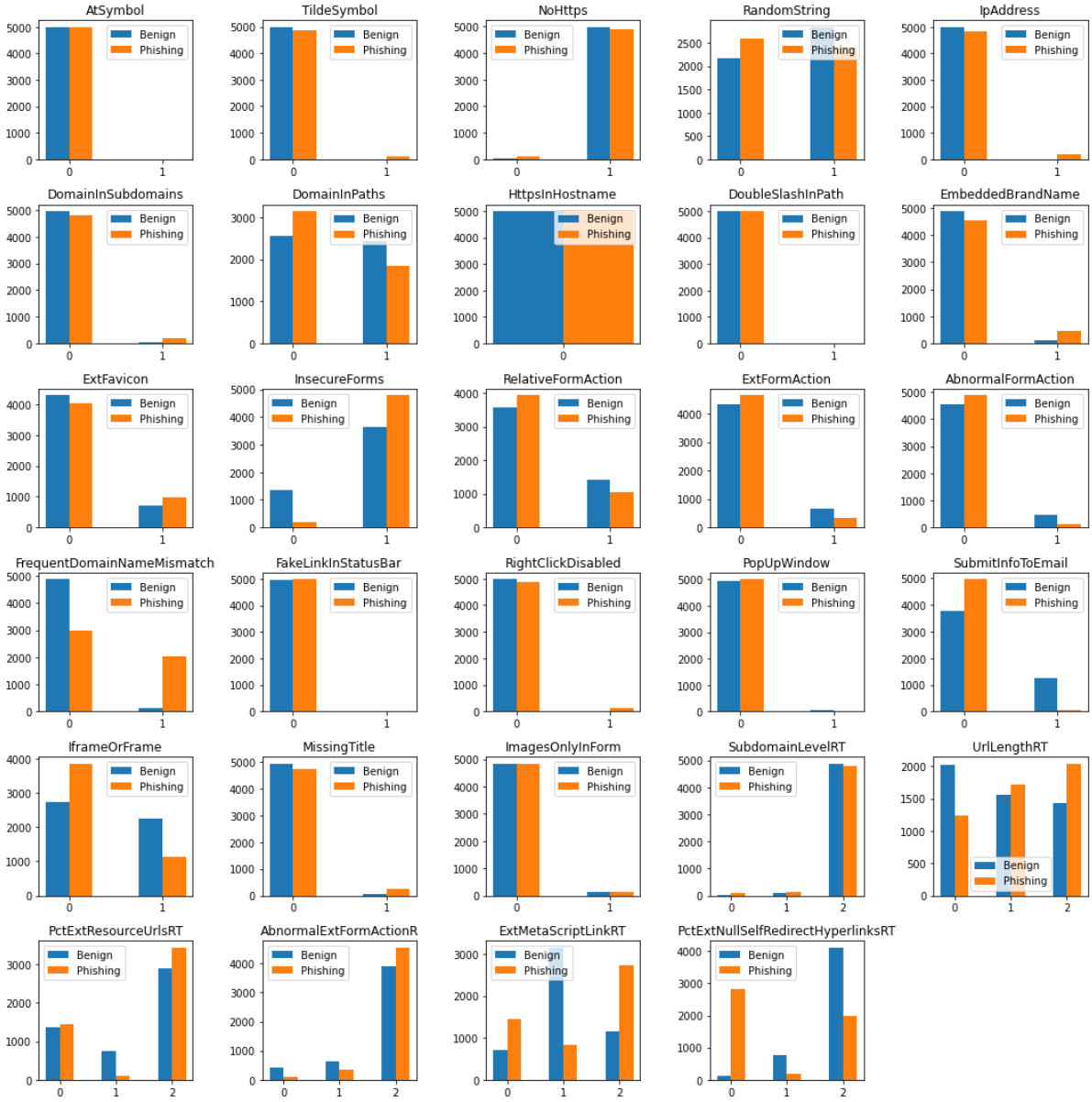


Figura 4.1: Relação dos atributos binários e categóricos com a classe.

Para exemplificar esta relação observada de alguns atributos com a classe, é possível destacar InsecureForms e SubmitInfoToEmail. O atributo InsecureForms indica se o formulário na página submete os dados para uma URL sem o protocolo HTTPS. O gráfico relacionado a este atributo indica que a presença deste mecanismo na página está mais presente em sites de phishing do que em sites benignos. Por outro lado, a presença da função HTML "mailto", observada no atributo SubmitInfoToEmail é uma característica ausente nos sites de phishing, mas pode ser encontrado em sites benignos.

As relações entre os atributos contínuos e discretos identificados na Tabela 3.1, e a classe, podem ser analisadas nos gráficos da Figura 4.2. O boxplot é uma tipo de gráfico que representa a variação dos dados observados de um atributo numérico por meio de quartis. Na análise dos atributos representados nestes gráficos, é possível destacar PctExtHyperlinks e PctExtResourceUrlsA, e em menor proporção, PctNullSelfRedirectHyperlinks. Nestes atributos, a probabilidade dos valores serem mais altos é maior

quando o site é phishing. De outra forma, e em proporção menor, nos valores do atributo PathLength, a probabilidade de se encontrar valores mais altos é maior quando o site é benigno.

Com o objetivo de exemplificar esta relação observada de alguns atributos com a classe, é possível destacar PctExtHyperlinks e PathLength. A análise do gráfico sobre o atributo PctExtHyperlinks indica que a porcentagem de hiperlinks externos no código HTML da página é maior nos sites de phishing. Por outro lado, ao observar o gráfico do atributo PathLength, é possível concluir que o total de caracteres na URL da página tende a ser maior em sites legítimos.

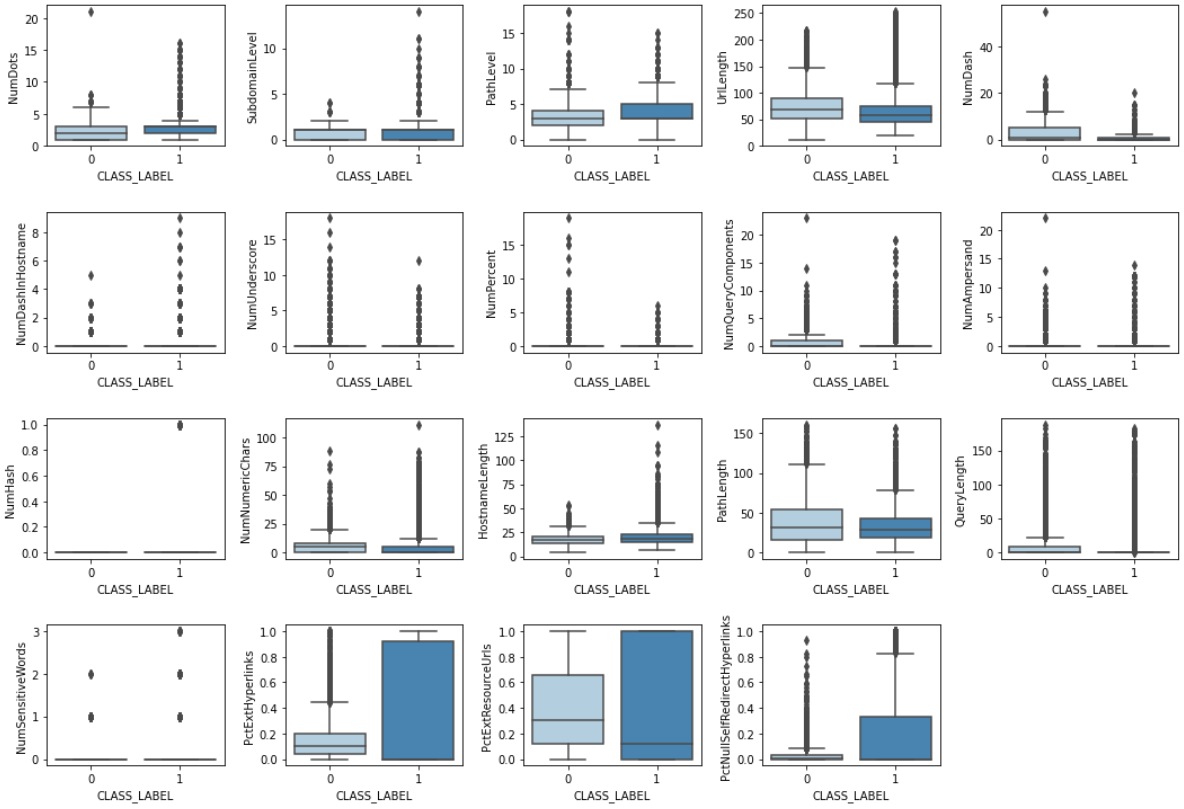


Figura 4.2: Relação dos atributos contínuos e discretos com a classe.

A relação entre os atributos é representada no mapa de calor da Figura 4.3, que mapeia os valores de uma matriz de correlação usando cores para representar os coeficientes de correlação entre os atributos, dispostos nos dois eixos. Os valores limítrofes indicam maior correlação entre os atributos, enquanto que valores próximos de zero indicam baixa correlação. Os maiores fatores de correlação foram observadas nos pares de atributos descritos na Tabela 4.1.

4.2 Treinamento e Avaliação

O conjunto de dados foi dividido em treinamento, avaliação e teste. Dos 10.000 registros contidos neste conjunto, foram extraídos 10% para o subconjunto de teste, ou seja, 1.000 registros. Dos 9.000 registros restantes, foram extraídos outros 10%, ou seja, 900 registros para o subconjunto de avaliação. Os 8.100 registros restantes foram utilizados no conjunto de treinamento. As divisões foram feitas de forma estratificada para garantir que a proporção de registros de sites benignos e phishing do conjunto de dados

Tabela 4.1: Destaque dos maiores fatores de correlação entre atributos.

Atributo 1	**Atributo 2**	**Fator de correlação**
AbnormalFormAction	AbnormalExtFormActionR	-0.818794
AbnormalExtFormActionR	AbnormalFormAction	-0.818794
PctExtResourceUrlsRT	PctExtResourceUrls	-0.804744
PctExtResourceUrls	PctExtResourceUrlsRT	-0.804744
UrlLengthRT	UrlLength	-0.800096
UrlLength	UrlLengthRT	-0.800096
PctExtResourceUrls	ExtMetaScriptLinkRT	-0.765659
ExtMetaScriptLinkRT	PctExtResourceUrls	-0.765659
QueryLength	NumAmpersand	0.754427
NumAmpersand	QueryLength	0.754427
QueryLength	NumQueryComponents	0.811784
NumQueryComponents	QueryLength	0.811784
NumAmpersand	NumQueryComponents	0.872951
NumQueryComponents	NumAmpersand	0.872951

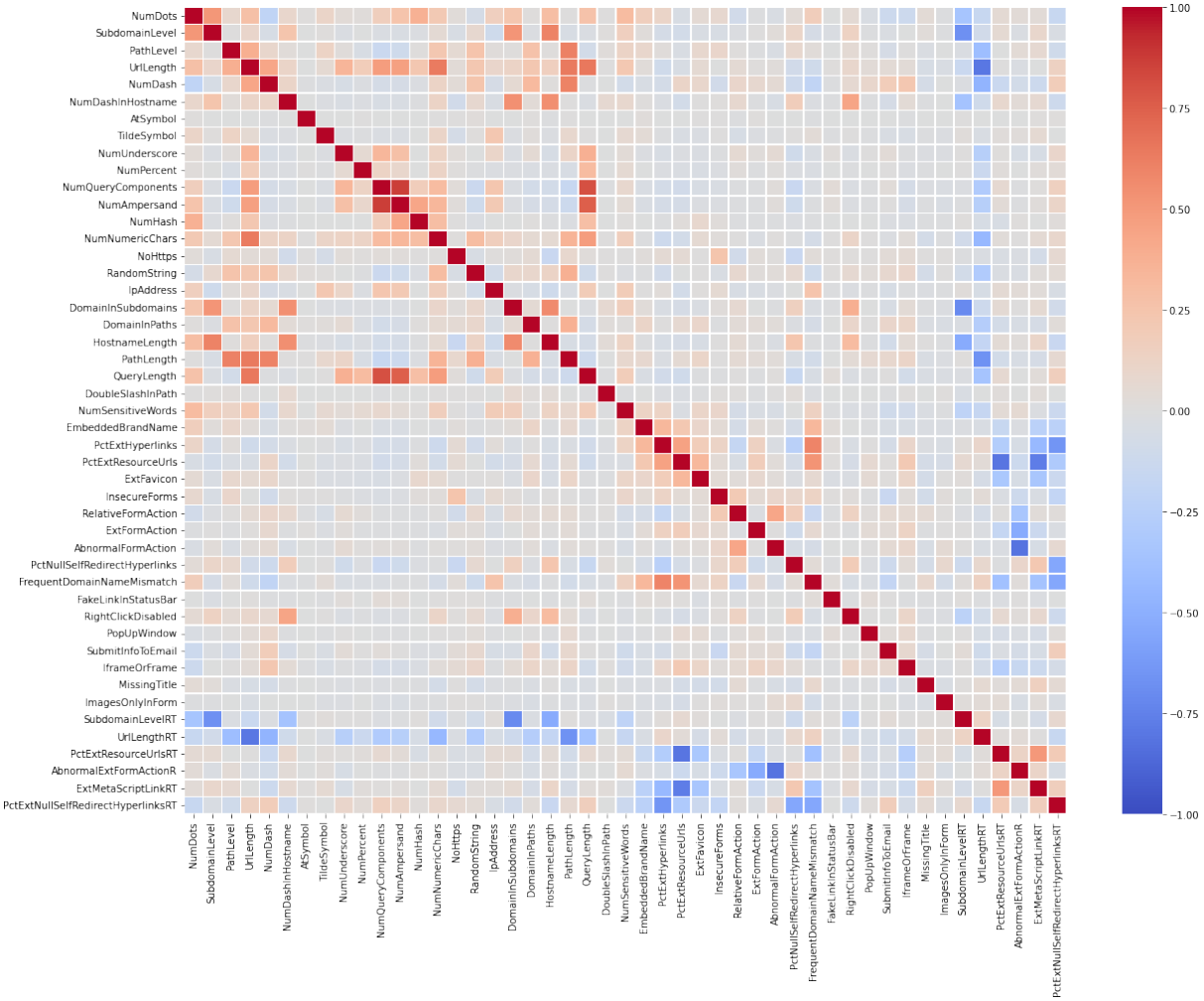


Figura 4.3: Mapa de correlação dos atributos.

original, que é de 50% para cada classe, fosse preservada nos subconjuntos gerados. Os dados também foram misturados antes das divisões.

O treinamento de cada modelo é realizado em 1.000 épocas. Uma época é uma iteração sobre todos os dados fornecidos, que incluem os 8.100 registros do subconjunto de treinamento e os 900 do subconjunto de validação. A atualização dos pesos da rede é feita após cada iteração com um determinado número de amostras do subconjunto de treinamento, chamado de lote. O tamanho dos lotes (*batch size*) foi definido como 8.100. Desta forma, os pesos da rede são atualizados uma vez a cada época, após a iteração com todas as amostras do subconjunto de treinamento. Os dados de validação são usados para avaliar a perda e a acurácia do modelo no final de cada época, e não são usados para treinar o modelo.

Nas Figuras 4.4 e 4.5 é possível observar a evolução da acurácia nos subconjuntos de treino e avaliação, respectivamente. A diferença de desempenho entre as MLPs e a RL é evidente, mas não é possível afirmar qual foi modelo obteve o melhor desempenho. O mecanismo para a escolha deste modelo foi construído da seguinte forma. Ao final do treinamento de cada modelo, as informações sobre a arquitetura e os pesos usados na época em que foi obtida a maior acurácia são armazenadas, permitindo que o melhor ajuste de cada modelo possa ser recuperado e reutilizado posteriormente nos experimentos da fase de testes.

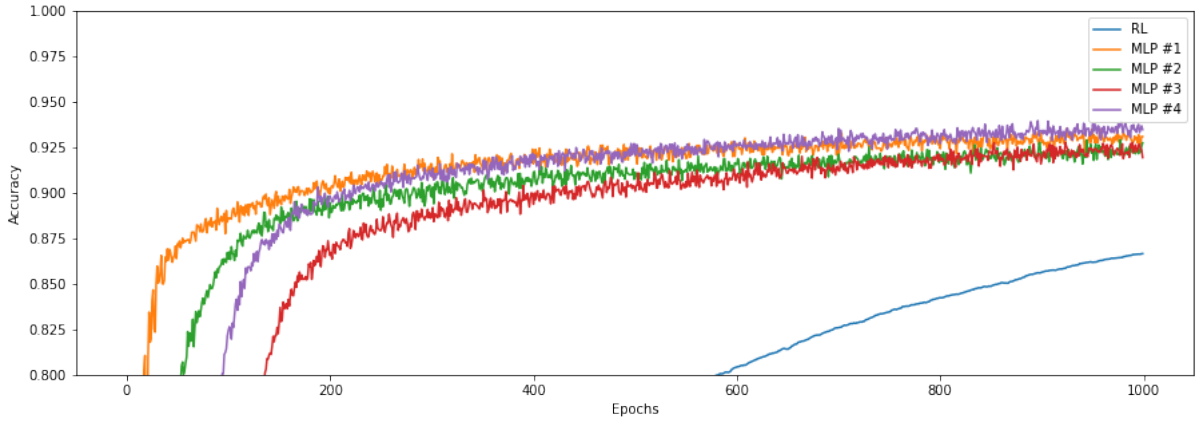


Figura 4.4: Evolução do desempenho dos modelos no treinamento para avaliação.

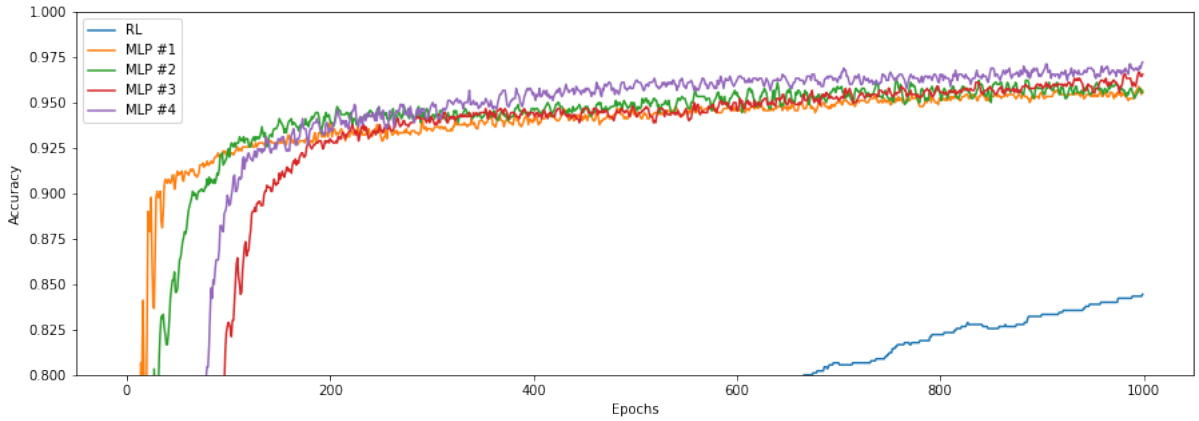


Figura 4.5: Evolução do desempenho dos modelos na avaliação.

A maior acurácia obtida por cada modelo no conjunto de validação é comparada, para que o modelo com melhor desempenho seja definido e recuperado para a fase de testes. O uso deste mecanismo é importante, pois a melhor acurácia de cada modelo não será necessariamente a obtida na última época. Os valores dos melhores desempenhos de cada modelo podem ser observados na Tabela 4.2.

Tabela 4.2: Desempenhos dos modelos na validação.

Modelo	**Acurácia**
RL	0.844
MLP #1	0.959
MLP #2	0.962
MLP #3	0.967
MLP #4	0.972

4.3 Testes

A MLP #4, que apresentou o melhor desempenho na avaliação é submetida aos testes, juntamente com o modelo RL. Nesta etapa, o subconjunto de avaliação é incorporado ao conjunto de treinamento. Dos 10.000 registros contidos no conjunto de dados, foram extraídos 10% para o subconjunto de teste, ou seja, 1.000 registros. Os 9.000 registros restantes foram utilizados no conjunto de treinamento do MLP com melhor desempenho e a RL. O tamanho dos lotes (*batch size*) foi redefinido para 9.000, de forma a se adequar ao tamanho do subconjunto de treinamento. Na Figura 4.6 é possível observar a evolução da acurácia no subconjunto de treino. Na Tabela 4.3 é possível visualizar os desempenhos dos modelos.

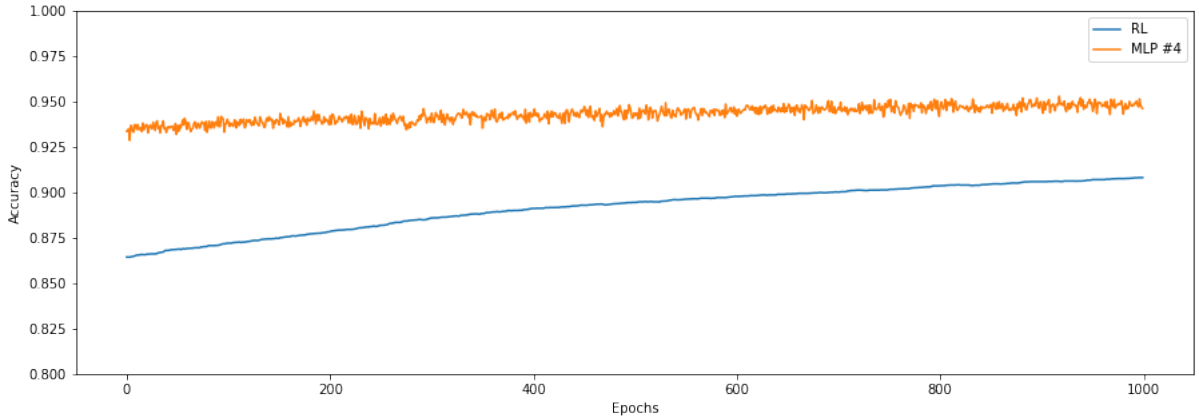


Figura 4.6: Evolução do desempenho dos modelos no treinamento para teste.

Tabela 4.3: Desempenhos dos modelos no teste.

Modelo	**Acurácia**
RL	0.905
MLP #4	0.972

Uma forma de comparar modelos de classificação binários é através da Curva ROC. O gráfico permite avaliar a variação da sensibilidade, representada pela taxa de verdadeiros positivos, e da especificidade, que é a taxa de falsos positivos, de acordo com o ponto de corte na probabilidade estimada de

um site ser phishing. Na Figura 4.7 é possível observar o gráfico da Curva ROC para comparação entre o desempenho da RL e do MLP #4 no conjunto de testes.

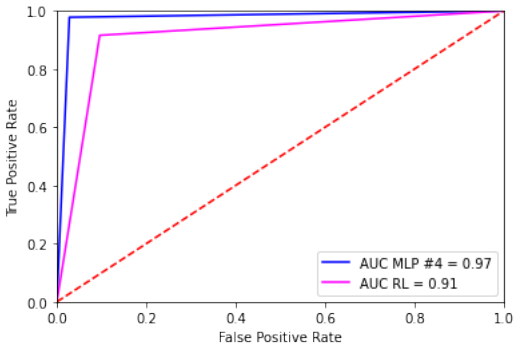


Figura 4.7: Curva ROC dos modelos RL e do MLP #4.

Capítulo 5

Conclusão

Neste trabalho, foi desenvolvido um classificador baseado em uma rede neural de aprendizado profundo com uma arquitetura MLP, mapeando as *features* extraídas no conjunto de dados disponibilizado por Tan (2018) para uma saída que identificou o site como *phishing* ou benigno. A partir da realização de experimentos, foi verificada que a rede neural de aprendizado profundo com quatro camadas ocultas obteve o melhor desempenho.

Foi verificado que, após a execução das etapas de treinamento e avaliação dos modelos, as redes com duas camadas ocultas ou mais, apresentaram desempenho semelhante. A realização de testes foi efetuada com a MLP com quatro camadas ocultas e seu desempenho comparado com um classificador baseado em regressão logística. Os resultados demonstraram que a MLP obteve melhor acurácia.

Como opção de evolução deste trabalho de pesquisa e aplicação prática de redes neurais profundas no tema da segurança da informação, é possível realizar novo experimento que seja comparável aos procedimentos realizados por Chiew et al. (2019). Outro ponto de evolução é experimentar outras arquiteturas, com cinco ou mais camadas ocultas, com diferentes quantidades de neurônios por camadas, de forma a avaliar o desepenho obtido nestes cenários.

Referências Bibliográficas

ABNT ISO/IEC 27002. *Tecnologia da Informação - Técnicas de Segurança - Código de Prática para a Gestão da Segurança da Informação*. [S.l.], 2013.

ALABDAN, R. Phishing attacks survey: Types, vectors, and technical approaches. *Future Internet*, v. 12, n. 10, 2020.

BAUM, E. B. On the capabilities of multilayer perceptrons. *J. Complex.*, Academic Press, Inc., USA, v. 4, n. 3, p. 193—-215, September 1988.

BUDUMA, N.; LOCASCIO, N. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2017. ISBN 1491925612.

CERT.br. *Cartilha de Segurança para Internet*. 2a. ed. São Paulo: Comitê Gestor da Internet no Brasil - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil, 2012.

CHIEW, K. L. et al. A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Inf. Sci.*, Elsevier Science Inc., USA, v. 484, p. 153–166, may 2019.

DAMODARAM, R. Study on phishing attacks and anti-phishing tools. *International Research Journal of Engineering and Technology (IRJET)*, n. 10, p. 700–705, January 2016.

HEATON, J. *Artificial Intelligence for Humans: Neural Networks and Deep Learning*. 1st. ed. [S.l.]: Heaton Research, Inc., 2015. v. 3.

HEBB, D. O. *The organization of behavior: A neuropsychological theory*. New York: Wiley, 1949. Hardcover.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: BENGIO, Y.; LECUN, Y. (Ed.). *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [S.l.: s.n.], 2015.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015.

MAYMI, F. J.; HARRIS, S. *CISSP All-in-One Exam Guide*. 8th. ed. [S.l.]: McGraw-Hill Education, 2018.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 127–147, 1943.

NIST SP 800-12. *An Introduction to Information Security*. Washington, D.C., 2017.

NIST SP 800-82. *Guide to Industrial Control Systems (ICS) Security*. Washington, D.C., 2015.

ROSENBLATT, F. *The perceptron - A perceiving and recognizing automaton*. Ithaca, New York, 1957.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. In: _____. *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988. p. 696—-699. ISBN 0262010976.

SHANKAR, A.; SHETTY, R.; NATH, B. A review on phishing attacks. *International Journal of Applied Engineering Research*, v. 14, n. 9, p. 2171–2175, 2019.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, JMLR.org, v. 15, n. 1, p. 1929–1958, jan 2014.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 7th. ed. [S.l.]: Pearson Education Limited, 2017.

TAN, C. L. *Phishing Dataset for Machine Learning: Feature Evaluation*. 2018. Data retrieved from Mendeley Data, <https://data.mendeley.com/datasets/h3cgnj8hft/1>.