APLICAÇÃO DE PERCEPTRON DE MÚLTIPLAS CAMADAS NA DETECÇÃO DE SPAM: REVISITANDO ESTRATÉGIAS COM INTELIGÊNCIA ARTIFICIAL CLÁSSICA

APPLICATION OF MULTI-LAYER PERCEPTRON IN SPAM DETECTION: REVISITING STRATEGIES WITH CLASSICAL ARTIFICIAL INTELLIGENCE

Edson Melo DE SOUZA
prof.edson.melo@gmail.com
Departamento de Informática
Universidade Nove de Julho - UNINOVE

Resumo
O crescimento do envio de e-mails não solicitados (spam) compromete a produtividade e a segurança em ambientes corporativos. Este estudo visa aplicar o modelo Perceptron de Múltiplas Camadas (MLP) para detectar mensagens de spam, utilizando um conjunto de 3.349 e-mails reais. Foram extraídos atributos simples dos campos "assunto", "remetente" e "cabeçalho", e a rede foi treinada com o algoritmo Backpropagation. Os resultados demonstraram acurácia de 91,13%, superando modelos como Naïve Bayes e AdaBoost. Mesmo com recursos computacionais modestos, o MLP apresentou desempenho competitivo, validando sua aplicação em cenários restritos. Conclui-se que o MLP é uma alternativa eficaz e didática para tarefas de classificação binária, sobretudo em contextos com limitações de infraestrutura e para fins educacionais em aprendizado de máquina.
Palavras-Chave
spam, redes neurais, inteligência artificial, correio eletrônico.

Abstract
The increasing volume of unsolicited email (spam) affects productivity and security in corporate settings. This study applies the Multi-Layer Perceptron (MLP) model to spam detection using a dataset of 3,349 real emails. Simple features from the "subject," "sender," and "header" fields were extracted, and the network was trained using the Backpropagation algorithm. Results showed 91.13% accuracy, outperforming models such as Naïve Bayes and AdaBoost. Despite limited computational resources, the MLP delivered competitive performance, validating its use in constrained environments. It is concluded that MLP remains an effective and educational alternative for binary classification tasks, especially in scenarios with infrastructural limitations or in introductory machine learning contexts where interpretability and simplicity are also important considerations.
Keywords
spam, neural networks, artificial intelligence, email.

INTRODUÇÃO

O correio eletrônico (e-mail) permanece como uma das principais ferramentas de comunicação no contexto corporativo e pessoal, devido à sua acessibilidade, baixo custo e abrangência global (SAHAMI *et al.*, 2021). No entanto, essas mesmas qualidades tornam o e-mail um alvo recorrente para o envio de mensagens não solicitadas em massa, conhecidas como spam. Este tipo de conteúdo compromete a produtividade dos usuários, sobrecarrega infraestruturas de rede e representa um risco significativo à segurança da informação, frequentemente associado à disseminação de *phishing*, malware e tentativas de fraude (FERRARA *et al.*, 2020).
Para combater o spam, uma variedade de abordagens tem sido proposta ao longo dos anos, incluindo técnicas baseadas em listas negras, heurísticas e, mais recentemente, algoritmos de

aprendizado de máquina (WEI; NGUYEN, 2022). Dentre os métodos clássicos de classificação, destacam-se os modelos Naïve Bayes, Support Vector Machines (SVM) e Redes Neurais Artificiais (RNAs). Embora técnicas modernas como os Transformers e RNAs profundas tenham alcançado desempenho superior em diversas tarefas, modelos mais simples continuam sendo relevantes, sobretudo em aplicações com restrições computacionais ou para fins educacionais.

Estudos recentes como o de Samarthrao e Rohokale (2022), destacam a eficácia de abordagens híbridas que combinam regras heurísticas com aprendizado de máquina para detecção de spam, enquanto soluções baseadas em BERT (DEVLIN *et al.*, 2019) são exploradas para classificação semântica avançada. No entanto, essas técnicas exigem recursos significativos, limitando sua aplicação em sistemas legados.

Este artigo apresenta um estudo de caso que aplica o Perceptron de Múltiplas Camadas (MLP) para detecção de spam em e-mails reais coletados em um ambiente corporativo. A proposta avalia a eficácia do modelo a partir de atributos sintáticos e semânticos simples extraídos dos campos do e-mail, buscando validar a capacidade do MLP em classificar mensagens como legítimas ou indesejadas com base em padrões reconhecíveis.

2 DEFINIÇÕES DE E-MAIL E SPAM

O E-mail ou Correio Eletrônico é um método que permite compor, enviar e receber mensagens através de sistemas eletrônicos de comunicação. O termo e-mail é aplicado tanto aos sistemas que utilizam a internet e são baseados no protocolo SMTP (*Simple Mail Transport Protocol*) (POSTEL, 1982), como aqueles sistemas conhecidos como intranets, que permitem a troca de mensagens dentro de uma empresa ou organização e são, normalmente, baseados em protocolos proprietários.

O envio e recebimento de uma mensagem de e-mail são realizados através de um sistema de correio eletrônico, o qual é composto de programas de computador que suportam a funcionalidade de gerir papéis de cliente e de um ou mais servidores de e-mail que, através de um endereço lógico – IP ou *Internet Protocol* – conseguem transferir uma mensagem de um usuário para outro. Estes sistemas utilizam protocolos que permitem o tráfego de mensagens de um remetente para um ou mais destinatários que possuem computadores conectados à internet.

O formato do e-mail está definido nas RFCs 2822, e 2045 a 2049 que são conhecidas como MIME. Mensagens de e-mail consistem basicamente de duas seções principais:

- **Cabeçalho (*header*)** – É estruturado em campos que contém os dados do remetente, destinatário, assunto e outras informações sobre a mensagem;
- **Corpo (*body*)** – Contém o texto da mensagem.
- O delimitador entre o corpo e o cabeçalho de uma mensagem é uma linha em branco.

Um e-mail típico pode ser visualizado na Figura 1.

Figura 1. Estrutura típica de um e-mail.

From: John Doe <markov@example.com>

Sender: Teddy Markov <mjones@machine.example>

To: Smith Anderson <smitha@example.net>

Subject: Festival

Date: Fri, 12 Nov 2004 04:52:06 -0600

Message-ID: <1234@local.machine.example>

This is a message just to say hello.

Pela facilidade de manipulação e baixo custo operacional, o e-mail facilita a disseminação do *spam*, termo usado para referir-se aos e-mails não solicitados (RAO *et al.*, 2021), que geralmente são enviados para um grande número de pessoas. Quando o conteúdo é exclusivamente comercial, esse tipo de mensagem é denominado de UCE (*Unsolicited Commercial E-mail*).
Dentre os problemas relacionados ao *spam* podem-se citar os conteúdos impróprios; prejuízos financeiros; não recebimento de mensagens legítimas; perda de tempo com lixo eletrônico e perda de produtividade. Já entre as modalidades de *spam* destacam-se:

- **Correntes** (*chain letters*): São os textos solicitando que a mensagem seja repassada para a sua lista de contatos;
- **Boatos** (*hoaxes*) e lendas urbanas: Mensagens com o objetivo de fomentar um determinado boato;
- **Propagandas**: Oferecer, sem autorização do destinatário, um produto ou serviço;
- **Códigos maliciosos** (*phishing*): Mensagens com o objetivo de roubar informações do usuário, como senhas, números de cartão de crédito etc.

Para que um e-mail seja classificado como *spam*, este deve apresentar características que possam diferenciá-lo das mensagens legítimas. A grande dificuldade encontrada nessa identificação está nos atributos que o compõe, pois, um e-mail legítimo pode ser entendido como *spam*, dependendo do método de classificação.
Entre as mensagens consideradas como *spam* e as consideradas legítimas, encontram-se os falso-positivos e os falsos-negativos. Os falsos-positivos são as mensagens legítimas classificadas como *spam*, já os falso-negativos são mensagens de *spam* classificadas como legítimas. A literatura apresenta diversas técnicas para detecção de spam, entretanto, o maior desafio é o de reduzir o número de falso-positivos, pois uma filtragem incorreta pode acarretar na perda de uma mensagem legítima.

2.1 Processo de Descoberta de Conhecimento e Mineração de Dados – KDD (*Knowledge Discovery and Data Mining*)

O processo de descoberta de conhecimento em bases de dados, conhecido como KDD (*Knowledge Discovery in Databases*), refere-se à extração de conhecimento útil, novo e compreensível a partir de grandes volumes de dados. Esse processo é composto por múltiplas etapas interativas e cíclicas, que vão desde a seleção e pré-processamento dos dados até a interpretação dos resultados extraídos por algoritmos de mineração (FAYYAD *et al.*, 1996; HAN *et al.*, 2022).
De forma geral, o KDD pode ser dividido em três grandes fases:

- **Pré-processamento**, onde os dados brutos são limpos, transformados e organizados em formatos compatíveis com as técnicas de mineração;
- **Mineração de dados**, na qual são aplicados algoritmos de aprendizado de máquina ou estatística para identificar padrões ou modelos relevantes;
- **Pós-processamento**, responsável por validar, interpretar e apresentar os resultados de forma compreensível, incluindo a visualização e a geração de relatórios (ZAKI; MEIRA JR., 2020).

A natureza iterativa do processo permite que, caso os resultados não atendam às expectativas, as fases anteriores sejam revisitadas para reformulação dos atributos ou escolha de novas abordagens. Essa característica é essencial em domínios dinâmicos como o combate ao spam, onde a evolução constante das técnicas de camuflagem exige adaptação contínua dos modelos preditivos (CHANDOLA *et al.*, 2009; DOU *et al.*, 2020).
No contexto da detecção de spam, o KDD se mostra especialmente relevante, pois permite identificar padrões linguísticos, comportamentais e estruturais em mensagens eletrônicas, que muitas vezes passam despercebidos em abordagens manuais ou heurísticas simples. A capacidade de extrair atributos discriminantes e ajustar modelos de forma automática contribui diretamente para

o aumento da eficácia de sistemas de filtragem inteligentes.

3 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados na estrutura e funcionamento do cérebro humano. Sua principal característica é a capacidade de aprender a partir de dados e generalizar esse conhecimento para novos exemplos, tornando-as eficazes em tarefas como classificação, regressão, previsão e reconhecimento de padrões (GOODFELLOW *et al.*, 2016).

Uma RNA é composta por unidades de processamento denominadas neurônios artificiais, mostrado na Figura 1, que são organizadas em camadas interconectadas. Cada neurônio realiza uma operação simples: recebe estímulos de entrada, pondera esses estímulos com base em pesos sinápticos, aplica uma função de ativação e transmite a saída para os neurônios da camada seguinte (HAYKIN, 2009).

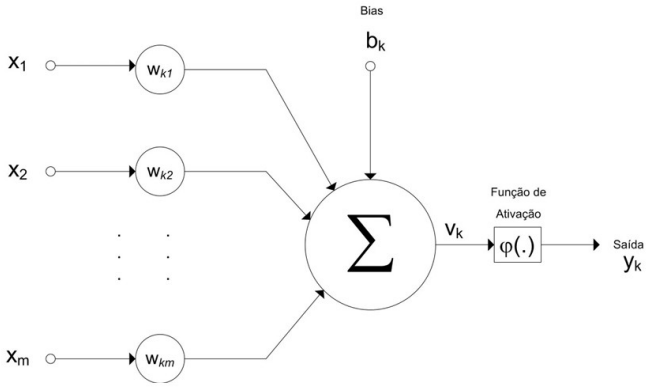
A capacidade de aprendizado das RNAs depende do ajuste adequado dos pesos dessas conexões. Durante o treinamento supervisionado, um conjunto de dados rotulado é utilizado para guiar o processo de correção dos erros da rede, permitindo que os pesos sejam ajustados iterativamente. O desempenho do modelo é validado com base em sua habilidade de gerar saídas corretas para dados não vistos anteriormente.

Diversos tipos de funções de ativação podem ser utilizadas para modelar o comportamento não linear dos neurônios, como a função logística (sigmoide), a tangente hiperbólica (*tanh*) e a ReLU (*Rectified Linear Unit*), esta última amplamente empregada em arquiteturas modernas por sua simplicidade computacional e eficácia no treinamento de redes profundas (GLOROT *et al.*, 2011; RIZO; CANATO, 2020).

Embora as RNAs profundas, como as redes convolucionais (CNNs) e as redes recorrentes (RNNs), tenham ampliado as fronteiras do aprendizado de máquina em áreas como visão computacional e processamento de linguagem natural, modelos clássicos como o Perceptron de Múltiplas Camadas (MLP) continuam sendo amplamente utilizados em tarefas de menor complexidade, em ambientes com restrições computacionais ou em aplicações educacionais (WEI; NGUYEN, 2020).

Neste contexto, a aplicação de um modelo MLP para detecção de spam oferece uma solução viável, eficaz e didática, ao mesmo tempo em que permite explorar conceitos fundamentais das redes neurais, como propagação direta, retropropagação do erro e ajuste dos pesos sinápticos.

Figura 2. Modelo de um Neurônio Artificial.



Fonte: HAYKIN, 2001.

Após o modelo de McCulloch-Pitts, surgiram outras abordagens, dando flexibilidade aos pesos e maior capacidade à Rede Neural; através dos neurônios com funções de ativação não-lineares, das arquiteturas com mais de uma camada e algoritmos apropriados para alterar os pesos sinápticos. De forma geral, nos neurônios artificiais, os seguintes elementos estão envolvidos:

- **Estímulos de entrada (*x*)**: Valores apresentados à rede;
- **Conjunto de sinapses (*w*)**: Ligações entre neurônios. Cada ligação possui um valor (peso),

que representa a sua força: os estímulos de entrada são multiplicados pelos respectivos pesos de cada ligação, podendo gerar um sinal tanto positivo (excitatório) quanto negativo (inibitório);

- **Combinador Linear (∑)**: Executa o somatório dos sinais produzidos pelo produto entre os pesos sinápticos e as entradas fornecidas ao neurônio. Em outras palavras, é o integrador dos sinais que chegam ao neurônio. A saída do neurônio é definida pelo seu valor de ativação calculado mediante a Equação 1:
- **Saída (*Y*)**: É o valor de resposta da rede.

$$v_k = \sum_{j=1}^{m} W_{kj} x_j + b_k \quad (1)$$

Equação (1)

Onde:
v é o campo local induzido do neurônio *k*;
w são os pesos das conexões do neurônio *k*;
x é o valor de cada um dos m estímulos que chegam ao neurônio ***k***;
b é o valor do bias que tem a função de aumentar ou diminuir a entrada líquida da função de ativação.

As Funções de Ativação fornecem o valor da saída de um neurônio em termos de campo local induzido.

3.1 Funções de Ativação

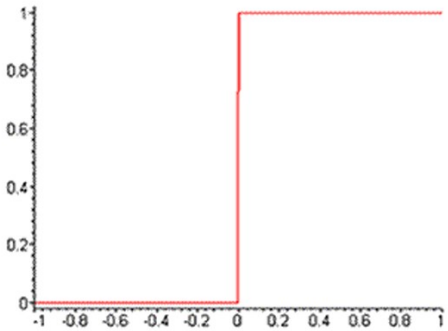
A Função Limiar ou *Heaviside* é referenciada na literatura para o modelo de McCulloch-Pitts. Ela possui a característica de "tudo-ou-nada", ou seja, assume o valor 1 se o campo local induzido de um neurônio é positivo, e 0 caso seja negativo. A Função Limiar é expressa por meio da Equação 2:

$$y_k = \begin{cases} 1, se\ v_k \geq 0; \\ 0, se\ v_k < 0; \end{cases}$$

Equação (2)

Nos neurônios construídos com essa função, a saída *y* será igual a 0, caso o valor de ativação *v* seja negativo e 1 nos casos em que o valor de ativação seja positivo. O limiar de ativação pode ser observado na Figura 3.

Figura 3: Função Limiar de Ativação. Fonte: LNCC.



A Função Sigmoide que, ao contrário da função limiar, pode assumir todos os valores entre 0 e 1 também pode ser utilizada como função de ativação. A sua representação mais utilizada é a função logística, definida pela Equação 3:

$$y_k = \frac{1}{1 + e(-av)}$$

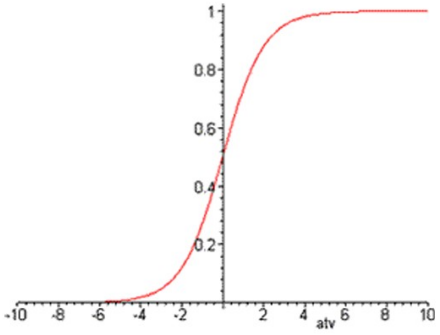
Equação (3)

Onde:

- a é o parâmetro de inclinação da função sigmoide;
- v é o valor de ativação do neurônio.

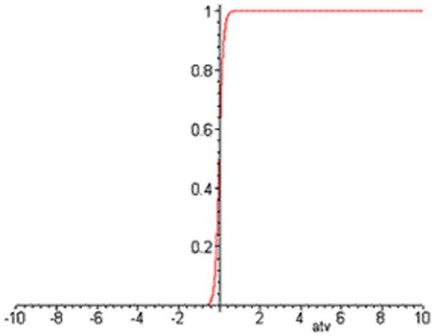
Na Figura 4 é mostrada a plotagem da função sigmoide.

Figura 4: Função Sigmoide. Fonte: LNCC.



Quando o valor do parâmetro é aumentado, tendendo ao infinito, esta função comporta-se como uma função de limiar, observada na Figura 5.

Figura 5: Função Sigmoide -a tendendo ao ∞. Fonte: LNCC.



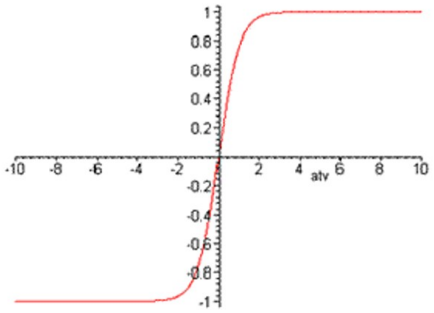
A Tangente Hiperbólica também pode ser utilizada como função de ativação. Como a função logística, também possui forma de "s", assumindo valores entre 1 e -1, sendo representada pela Equação 4, onde v é o valor de ativação da unidade.

$$\delta(v) = \tanh(v)$$

Equação (4)

Na Figura 6 pode-se visualizar o comportamento da Função Tangente Hiperbólica.

Figura 6: Função Tangente Hiperbólica.



Fonte: LNCC.

3.2 Processo de Aprendizagem em Redes Neurais

Para que uma RNA possa fornecer resultados convenientes é necessário que a mesma passe por uma fase de treinamento, onde seus pesos são ajustados de forma que ela se adapte aos diferentes estímulos de entrada. Durante a fase de treinamento, ocorre o seu aprendizado. Há vários processos de aprendizado, os quais, de forma geral, podem ser classificados em:

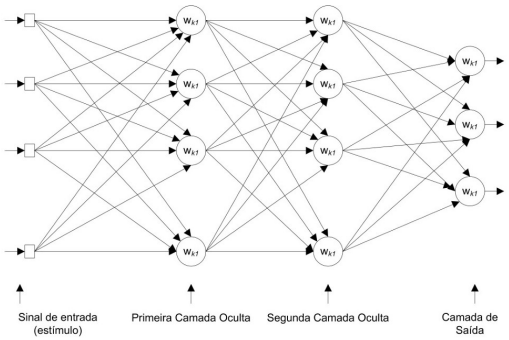
- **Aprendizado Supervisionado**: É fornecida uma referência do objetivo a ser alcançado. Um exemplo de processo de Aprendizado Supervisionado é o Aprendizado por Correção de Erro.
- **Aprendizado Não-Supervisionado**: Neste caso não é fornecida nenhuma referência externa. Podemos citar como exemplo o processo de Aprendizado Competitivo e o Aprendizado Hebbiano.

Entre os diversos modelos de RNAs, o do tipo Perceptron foi o pioneiro nesta área. Simples e de fácil implementação, utiliza neurônios do tipo McCulloch Pitts. Limitada à classe de problemas linearmente separáveis, pode, no entanto, ser utilizada em tarefas de classificação simples.
O Perceptron, proposto por Rosenblatt, é composto pelo neurônio de McCulloch Pitts, com Função de Limiar e Aprendizado Supervisionado (ROSENBLAT; PAPERT, 2021). Sua arquitetura consiste em uma camada de entrada e uma camada de saída. A limitação desta RNA se encontra na reduzida gama de problemas que consegue tratar, ou seja, na classificação de conjuntos linearmente separáveis.

3.3 Modelo Perceptron de Multi-Camadas

O Perceptron de Multi-Camadas é uma extensão do Perceptron simples, capaz de trabalhar com problemas não-linearmente separáveis (RAMCHOUN *et al.*, 2016). Este avanço foi possível através da utilização de, pelo menos, uma camada entre a camada de entrada e a camada de saída. Estas camadas intermediárias, conhecidas como camadas ocultas, trabalham como um reconhecedor de características, que ficam armazenadas nos pesos sinápticos.
O algoritmo de treinamento mais utilizado para esse modelo é o "Backpropagation" (LILLICRAP *et al.*, 2020), um tipo de Aprendizado Supervisionado por Correção de Erro. A modelagem da arquitetura de uma rede Perceptron Multi-Camadas, Figura 7, envolve a escolha da quantidade de camadas e o número de unidades em cada camada. Destacam-se para esse modelo alguns aspectos importantes: (i) escolha do número de unidades de entrada; (ii) definição da função de ativação que irá direcionar o comportamento da rede; (iii) codificação da camada de saída e a formatação da resposta da rede.

Figura 7: Grafo arquitetural de um perceptron de múltiplas camadas.



Fonte: HAYKIN, 2001.

Outros parâmetros devem ser escolhidos, referentes ao treinamento tais como taxa de aprendizado e o conjunto de treinamento. A respeito do conjunto de treinamento devem ser estudados os dados relevantes, os quais destaquem as características que devem realmente ser "aprendidas" pela rede. O processamento de cada unidade é influenciado pelo processamento efetuado pelas unidades das camadas anteriores. Cada camada desempenha um papel específico, como segue:

- **Camada de Entrada**: Receptora de estímulos;
- **Primeira Camada Oculta**: Cada unidade desta camada define uma reta no espaço de decisão, refletindo características dos padrões apresentados;
- **Segunda Camada Oculta**: Combina as retas definidas pela camada anterior, formando regiões convexas onde o número de lados é definido pelo número de unidades da camada anterior, conectados a unidade desta camada;
- **Camada de Saída**: Combina as regiões formadas pela camada anterior, definindo o espaço de saída da rede.

As camadas intermediárias da rede são como detectores de características, as quais serão representadas, internamente, através dos pesos sinápticos.

3.4 Algoritmo Backpropagation

Backpropagation é o algoritmo para treinamento de redes Multicamadas mais difundido, sendo constituído de duas fases: propagação e retropropagação (LILLICRAP *et al.*, 2020).
Na fase de propagação, após ser apresentado o padrão de entrada, a resposta de uma unidade é propagada como entrada para as unidades na camada seguinte, até a camada de saída, onde é obtida a resposta da rede e o erro é calculado. Na retropropagação os pesos são recalculados desde a camada de saída até a camada de entrada, ou seja, no sentido contrário da apresentação. Durante a fase treinamento apresenta-se um conjunto formado pelo par "entrada para a rede" e "valor desejado para resposta" a entrada. A saída será comparada ao valor desejado e será computado o erro global da rede, que influenciará na correção dos pesos no passo de retropropagação, apesar de não haver garantias que a rede forneça uma solução ótima para o problema.
Os passos do algoritmo Backpropagation são descritos a seguir:

1- **Inicialização**: Inicializa os pesos sinápticos e o bias (b) aleatoriamente, com valores no intervalo [-1, 1];
2- **Apresentação dos padrões de treinamento**: Treinamento "on-line" - Para cada exemplo do conjunto de treinamento, efetue os passos 3 e 4. Treinamento "em lote": Para cada "época" do conjunto de treinamento, efetue os passos 3 e 4.
3- **Computação adiante (Propagação)**: Depois de apresentado o exemplo do conjunto de treinamento T = $\{x(n), d(n)\}$, sendo $x(n)$ a entrada apresentada à rede e $d(n)$ a saída desejada, calcula o valor da ativação v_j por meio da Equação 5:

$$y_k = \frac{1}{1+e(-av)}$$

Equação (5)

Para o cálculo da saída *y* da unidade *k*, utilizando a função sigmoide, como no exemplo, utilize a saída das unidades de uma camada como entradas para a seguinte, até a última camada. A saída das unidades da última camada será a resposta da rede.

4- **Calcula o Sinal de Erro**: Fazendo a saída $y_j = O_j(n)$, será $O_j(n)$ a resposta da rede. Calcule o sinal de erro de acordo com a Equação 6:

$$e_j(n) - d_j(n) - O_j(n)$$

Equação (6)

Onde: $d_j(n)$ é a saída desejada com resposta para cada unidade na interação (*n*). Este sinal de erro será utilizado para computar os valores dos erros das camadas anteriores e fazer as correções necessárias nos pesos sinápticos.

5- **Computação para trás (Retropropagação)**: Calcule os erros locais, *d*, para cada unidade, desde a camada de saída até a de entrada. O gradiente local é definido pelas equações 7 e 8.

$$\delta_j(n) = e_j(n)O_j(n)(1-O_j(n))$$

Equação (7)

Para unidade da camada de saída ou,

$$\delta_j(n) = O_j(n)(1-O_j(n))\sum \delta_k W_{kj}$$

Equação (8)

Para as unidades das demais camadas, onde:

- $O_j(i\text{-}O_j)$ é a função de ativação diferenciada em função do argumento. Valor de ativação;
- δ_k é o erro das unidades da camada anterior, conectadas a unidade;
- w_{kj} são os pesos das conexões com a camada anterior.

Após o cálculo dos erros de cada unidade, calcule o ajuste dos pesos de cada conexão segundo a regra delta generalizada e atualize os pesos segundo a Equação 9:

$$\Delta w_{kj}(n+1) = \alpha w_{kj}(n) + \eta \delta_j y_j$$

Equação (9)

Para os cálculos dos ajustes dos pesos faça, Equação 10:

$$w(n+1) = w(n) + \Delta w_{kj}(n)$$

Equação (10)

Para atualizar os pesos sinápticos onde:

- α - é a constante de momento, quando α=0, esta função funciona como a regra delta comum;
- η- é a taxa de aprendizado;
- y_j- é a saída produzida pela unidade *j*.

6- Interação: Refaça os itens 3, 4 e 5 referentes à propagação, cálculo do erro e

retropropagação, apresentando outros estímulos de entrada, até que sejam satisfeitas as condições de treinamento; as quais podem ser: o erro da rede está baixo, sendo pouco alterado durante o treinamento; o número máximo de ciclos de treinamento foi alcançado.

3.5 Correção de Erro

O erro de uma Rede Neural pode ser calculado como a diferença entre a saída real gerada pela rede e a saída desejada, fornecida em um ensino supervisionado a partir da equação 11:

$$e_k = d_k - y_k$$

Equação (11)

Onde, para um estímulo *k*:

- e_k – sinal de erro;
- d_k – saída desejada apresentada durante o treinamento;
- y_k – saída real da rede após a apresentação do estímulo de entrada.

Durante o aprendizado supervisionado, portanto, os erros vão sendo calculados sucessivamente, até que cheguem a um valor satisfatório, definido a priori. Sendo assim, surge uma curva de erros, a qual está diretamente relacionada à natureza do modelo de neurônio utilizado.
Se a rede é formada por unidades lineares, como no modelo de McCulloch-Pitts, na superfície de erro será encontrada um único valor mínimo. Por outro lado, se a rede é constituída por unidades não-lineares, podem ser encontrados diversos valores mínimos chamados de mínimos locais, além do mínimo global.
O processo de Aprendizado por Correção de Erros utiliza algoritmos para caminhar sobre a curva de erros, com o intuito de alcançar o menor valor de erro possível, o mínimo global. Muitas vezes, o algoritmo não alcança este mínimo global, atingindo o que chamamos de mínimo local. Caso este erro alcançado seja desfavorável, é necessário recomeçar o processo de aprendizado.
Para a correção do erro, os pesos da rede devem ser ajustados, de forma a aproximar a saída real à desejada. De acordo com a Regra Delta de Aprendizado, apresentada a seguir, tal ajuste dependerá do próprio erro calculado; do valor do estímulo de entrada que é "transmitido" pelo peso a ser ajustado; e também da taxa de aprendizado, a qual se relaciona à cautela com que a curva de erros é percorrida. Para um dado estímulo *k*, no passo de treinamento *n*, aplica-se a Equação 12:

$$\Delta w_{ji} = \eta e_k(n) x_j(n)$$

Equação (12)

Onde:

- $\Delta w_{ji}(n)$ – Valor de ajuste a ser acrescido ao peso w_{ji};
- η – Taxa de aprendizado;
- $e_k(n)$ – Valor do erro;
- $x_j(n)$ – Valor do estímulo.

O valor do peso será calculado como descrito na Equação (13).

$$w(n+1) = w(n) + \Delta w_{kj}(n)$$

Equação (13)

Portanto, pode-se utilizar a Regra Delta para corrigir os valores dos pesos, minimizando a função de erro e, também conhecida como Função de Custo, Equação 14:

$$\varepsilon(n) = \frac{1}{2} e^2(n)$$

Equação (14)

Onde:

- ε (*n*) é o erro da rede no passo *n* do treinamento;
- e^2 é o valor da função de custo no passo *n* do treinamento.

4 MATERIAS E MÉTODOS

Para o desenvolvimento do trabalho foram selecionadas 5.437 mensagens de correio eletrônico em uma empresa de médio porte com a colaboração de 9 funcionários, onde, após uma filtragem manual por relevância e exclusão de referências cruzadas, obteve-se um total de 3.349 mensagens sendo 1.645 consideradas legítimas e 1.704 como spam.
Embora o conjunto de dados utilizado neste estudo (3.349 e-mails) seja limitado em comparação com bases públicas maiores, como o dataset Enron-Spam (aproximadamente 50.000 mensagens), ele reflete características específicas de ambientes corporativos reais. Essa abordagem é válida para cenários onde a coleta de dados é restrita devido a políticas de privacidade ou recursos computacionais limitados. Trabalhos futuros explorarão a ampliação da base de dados com mensagens públicas e técnicas de *data augmentation* (MAHARANA *et al.*, 2022) para validar a generalização do modelo.
Os algoritmos de extração e tratamento dos dados foram inicialmente desenvolvidos em linguagem ANSI C, utilizando o compilador GNU/GCC e a biblioteca REGEX em ambiente Linux/Ubuntu. Essa escolha visou garantir maior controle sobre a manipulação textual e sobre o desempenho computacional durante a fase de pré-processamento. Contudo, vale destacar que atualmente existem bibliotecas de alto nível, como *scikit-learn*, *TensorFlow* e *PyTorch*, que oferecem maior produtividade e simplicidade na implementação de Redes Neurais Artificiais (LIU, 2020). O uso de tecnologias mais recentes pode facilitar a reprodutibilidade e a expansão deste estudo, tornando-o mais acessível para fins educacionais e profissionais.
Através do KDD, foram extraídas as palavras do campo “Assunto” para compor uma lista de restrições. Tais palavras foram elencadas de acordo com o grau de aparição nas mensagens apontadas como *spam*. Na segunda etapa foram extraídos os endereços de e-mail com base na RFC 822 [*Request for Comments*: 822, 1982], seguidos da verificação da presença de imagens contidas no *header* da mensagem.

4.1 Pré-processamento dos Atributos

Após a análise dos dados brutos obtidos, estes foram submetidos a um algoritmo que efetuou o tratamento e a separação dos caracteres alfanuméricos por meio de uma lista definida pela relevância dos mesmos. Os caracteres selecionados foram:
{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,x,z,y,z,-,_}

Para obter-se uma boa performance e homogeneidade dos dados, todos os caracteres foram convertidos para caixa baixa, objetivando um menor esforço do algoritmo no processamento.
Após esta fase, foram efetuadas análises estatísticas a fim de consolidar os padrões a serem utilizados nos estudos com a RNA. Determinou-se que a utilização de percentuais da ocorrência de aparição dos caracteres sobre o total de caracteres que compõe o endereço do e mail, assim como o campo “Assunto”, era de grande relevância para o estudo. Desta forma, foram efetuados os cálculos dos percentuais através de um algoritmo aplicado sobre os dados brutos conhecidos. Tais informações foram obtidas mediante a aplicação da Equação 15:

$$f(c_i) = \frac{\sum_{i=1}^{t} c_i}{T}$$

Equação (15)

Onde:

- *f* é o percentual do *i-ésimo* caractere sendo analisado;
- c_i é o *i-ésimo* caractere analisado;
- T = t é o total de caracteres do endereço de e-mail ou do assunto.

A partir dos percentuais obtidos pela Equação 15, foi possível qualificar os atributos para compor a base de dados. São eles:

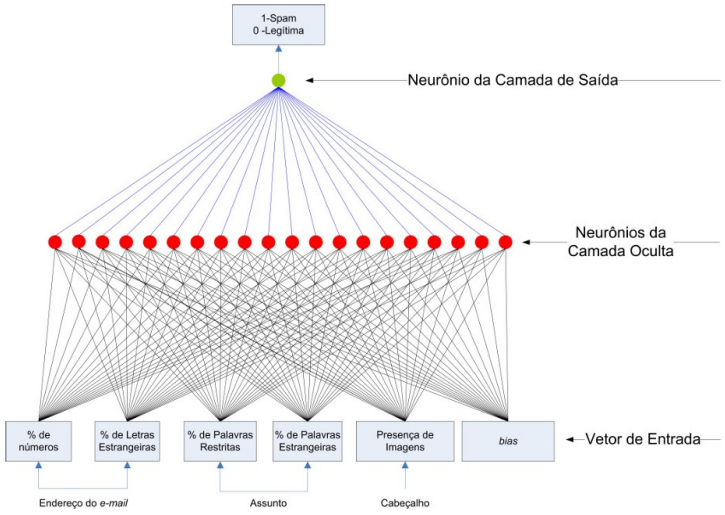
- Atributos sobre o endereço do e-mail
  - 1 - Percentual de números;
  - 2 - Percentual de letras estrangeiras;
  - 3 - Percentual de palavras restritas.
- Atributos sobre o header do e-mail
  - 4 - Presença de Imagem (atribui-se o valor 0.12, caso encontrada);
- Atributos sobre o assunto do e-mail
  - 5 - Percentual de palavras estrangeiras;
- Atributo de Qualificação
  - 6 - Classificações da mensagem (0 para normal e 1 para spam).

O próximo passo foi efetuar os testes através de uma RNA supervisionada MLP (*Multi Layer Perceptron*) treinada com o algoritmo Backpropagation.

4.2 Arquitetura MLP Utilizada

Para o desenvolvimento do trabalho, foi adotado o modelo MLP. Este modelo destaca-se entre os outros clássicos reconhecidos pela literatura por sua implementação ser relativamente fácil e seus resultados geralmente demonstrarem ser satisfatórios.
O treinamento do MLP foi realizado com os seguintes hiperparâmetros: taxa de aprendizado de 0,002, função de ativação sigmoide nas camadas ocultas e saída, e validação cruzada com 10 *folds*. Os pesos sinápticos foram inicializados aleatoriamente no intervalo [-1, 1], e o processo foi interrompido quando o erro de validação não diminuía por 20 épocas consecutivas (*early stopping*).
A arquitetura final (6 neurônios de entrada, 20 na camada oculta e 1 na saída) foi selecionada após testes com diferentes configurações, priorizando a relação entre complexidade e desempenho.

Figura 8: Arquitetura MLP utilizada. Fonte: O autor.



Conforme pode ser visto na Figura 8, o vetor de entrada é composto por 6 elementos. Esses elementos são os 5 atributos extraídos dos e-mails os quais podem qualificar um e-mail como *spam* ou legítimo, além do bias.

4.3 Treinamento da Rede MLP

O conjunto de dados obtido através do pré processamento foi submetido à RNA para treinamento e aprendizagem da rede em 3 situações diferentes, ou seja, os valores estipulados para o erro nos treinamentos foram: 0.001, 0.002 e 0.003. Os resultados dos treinamentos são apresentados na Tabela 1. Para o treinamento da rede foram utilizadas 1.704 mensagens sem duplicidades internas e ajustadas para submissão. O tempo de treinamento para o valor 0.001 foi de 2 minutos. Para os outros dois, 6 e 8 segundos, respectivamente. Para o erro 0.001, o tempo gasto não apresentou relevância, pois o processo foi efetuado apenas 1 vez, como é de característica das RNAs.

Tabela 1: Convergência em épocas.

Erro Inicial	Épocas	Erro Final
0.001	4.589	0.000769
0.002	113	0.001953
0.003	7	0.002917

5 RESULTADOS E DISCUSSÕES

Após o treinamento da Rede Neural Artificial com o algoritmo Backpropagation, o modelo foi avaliado utilizando um novo conjunto de 1.645 mensagens previamente classificadas como legítimas ou spam. O mesmo processo de pré-processamento aplicado aos dados de treinamento foi replicado neste conjunto de teste, garantindo consistência na entrada dos atributos.

Os resultados obtidos demonstraram desempenho satisfatório, com alta taxa de acerto na classificação das mensagens. A média de erro quadrático manteve-se estável ao longo das execuções, mesmo sem convergência absoluta em todas as instâncias, o que é aceitável em modelos com estruturas simples como o MLP. O tempo médio de resposta para a classificação de cada mensagem variou entre 0,1 e 0,2 segundos, indicando viabilidade para aplicações com demanda em

tempo real, desde que com volume moderado de requisições.

A Tabela 2 apresenta a comparação da acurácia do modelo MLP com algoritmos clássicos de aprendizado supervisionado amplamente utilizados na literatura, como Naïve Bayes, AdaBoost e C4.5. Observa-se que o MLP alcançou uma taxa de acerto de 91,13%, superando os modelos Naïve Bayes (89,74%) e AdaBoost (82,85%), e ficando atrás apenas do C4.5 (96,11%). Em relação aos erros de classificação, o MLP apresentou 4,10% de falsos positivos e 4,77% de falsos negativos.

Esses resultados confirmam que, mesmo diante de abordagens mais sofisticadas disponíveis atualmente, como árvores de decisão otimizadas ou redes neurais profundas, a arquitetura MLP continua sendo uma opção eficaz para tarefas de classificação binária simples, especialmente quando se deseja um equilíbrio entre desempenho, interpretabilidade e baixo custo computacional (WEI; NGUYEN, 2020).

Tabela 2. Resultados Obtidos com MLP *vs.* Literatura.

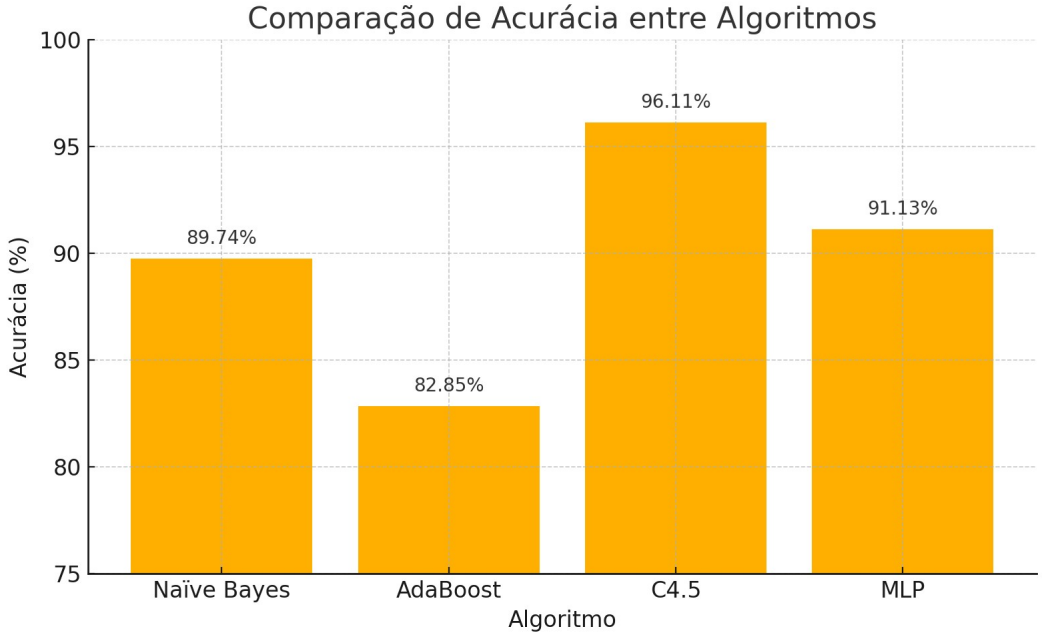
Algoritmo	**Acurácia**	**Precision (Spam)**	**Recall (Spam)**	**F1-Score (Spam)**	**Falso Positivo**	**Falso Negativo**
Naïve Bayes	89,74%	0,88	0,85	0,86	-	-
AdaBoost	82,85%	0,81	0,79	0,80	-	-
C4.5	96,11%	0,97	0,96	0,96	-	-
MLP	91,13%	0,92	0,90	0,91	4,10%	4,77%

Fonte: WALLACE e NIEVOLA, 2004.

Além da acurácia, métricas como Precision, recall e F1-Score foram calculadas para avaliar o desempenho do MLP em detalhes, mostrados na Tabela 2. O modelo apresentou um F1-Score de 0,91 para a classe spam, indicando equilíbrio entre precisão e recall. No entanto, a taxa de falsos negativos (4,77%) permanece superior à de falsos positivos (4,10%), destacando a necessidade de ajustes futuros para reduzir a liberação de mensagens indesejadas. Apesar da classificação dos falso-positivos e falso-negativos estarem acima dos outros algoritmos utilizados, ainda assim o resultado é expressivo em vista ao tamanho da amostra e simplicidade de implementação.

Na Figura 9 é mostrada a comparação entre a acurácia dos algoritmos Naïve Bayes, AdaBoost, C4.5 e MLP, com base nos resultados do estudo.

Figura 9. Comparação entre a acurácia dos algoritmos Naïve Bayes, AdaBoost, C4.5 e MLP.



6 CONSIDERAÇÕES FINAIS

Este estudo apresentou a aplicação de um modelo de Perceptron de Múltiplas Camadas (MLP) na tarefa de detecção de spam em e-mails, utilizando atributos simples extraídos de mensagens reais coletadas em ambiente corporativo. A proposta buscou validar a eficácia do MLP em cenários reais com restrições de dados, como ambientes corporativos onde a coleta de grandes volumes de e-mails é limitada por políticas de privacidade.

A arquitetura adotada demonstrou ser eficaz, com boa taxa de acerto na classificação das mensagens, mesmo com um conjunto limitado de variáveis e recursos computacionais modestos. O treinamento supervisionado com algoritmo Backpropagation foi capaz de ajustar os pesos sinápticos da rede de forma eficiente, permitindo ao modelo aprender padrões relevantes na diferenciação entre mensagens legítimas e mensagens de spam. Os resultados obtidos foram comparáveis, e em alguns casos superiores, aos de métodos tradicionais como Naïve Bayes e AdaBoost, conforme indicado na literatura.

Embora técnicas mais avançadas de aprendizado profundo, como redes convolucionais e modelos baseados em transformadores, ofereçam maior precisão em cenários complexos, o MLP continua sendo uma alternativa válida para aplicações com restrições de processamento ou voltadas ao ensino de fundamentos em aprendizado de máquina.

Entre as limitações deste estudo, destaca-se o tamanho reduzido do conjunto de dados utilizado, o que pode restringir a generalização dos resultados obtidos. Para pesquisas futuras, recomenda-se a utilização de bases de dados mais amplas e representativas, como o Enron-Spam Dataset ou coleções de mensagens provenientes de plataformas corporativas, obtidas mediante consentimento, a fim de avaliar a robustez do modelo em diferentes contextos.

Como direções para trabalhos futuros, sugere-se a expansão do corpus de mensagens, a incorporação de atributos semânticos extraídos do conteúdo textual e a experimentação com arquiteturas de rede mais sofisticadas. Ademais, a aplicação de técnicas híbridas de seleção de atributos, aliadas a métodos de ensemble learning, poderá fortalecer a abordagem proposta, contribuindo para o desenvolvimento de modelos mais eficazes na detecção automatizada de mensagens indesejadas.

REFERÊNCIAS BIBLIOGRÁFICAS

DEVLIN, J.; CHANG, M. W.; LEE, K.; TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, v. 1, p. 4171-4186, jun. 2019. DOI: 10.48550/arXiv.1810.04805.

DOU, Y.; MA, G.; YU, P. S.; XIE, S. Robust spammer detection by Nash reinforcement learning. In: *ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING (KDD)*, 2020. Anais [...], 2020. DOI: 10.48550/arXiv.2006.06069.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. *AI Magazine*, v. 17, n. 3, p. 37–54, 1996.

FERRARA, E.; VAROL, O.; DAVIS, C.; MENCZER, F.; FLAMMINI, A. The rise of social bots. *Communications of the ACM*, v. 62, n. 6, p. 96-104, 2020. DOI: 10.1145/2818717.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: *INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS*, 15., 2011. Anais [...]. [S.l.: s.n.], 2011. p. 315–323. Disponível em: https://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf. Acesso em: 1 mai. 2025.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. Cambridge: MIT Press, 2016.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. 4. ed. Cambridge: Morgan Kaufmann, 2022.

HAYKIN, S. *Neural networks and learning machines*. 3. ed. New Jersey: Pearson, 2009.

LILLICRAP, Timothy P.; SANTORO, Adam; MARRIS, Luke; AKERMAN, Colin J.; HINTON,

Geoffrey. Backpropagation and the brain. *Nature Reviews Neuroscience*, v. 21, n. 6, p. 335-346, 2020. DOI: 10.1038/s41583-020-0277-3.
LIU, Yuxi Hayden. *Python machine learning by example: build intelligent systems using Python, TensorFlow 2, PyTorch, and Scikit-Learn*. Birmingham: Packt Publishing Ltd, 2020.
MAHARANA, Kiran; MONDAL, Surajit; NEMADE, Bhushankumar. A review: data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, v. 3, n. 1, p. 91-99, 2022. DOI: 10.1016/j.gltp.2022.04.020.
RAMCHOUN, Hassan; GHANOU, Youssef; ETTAOUIL, Mohamed; IDRISSI, Mohammed Amine Janati. Multilayer perceptron: architecture optimization and training. 2016. DOI: 10.1145/3090354.3090427.
RAO, S.; VERMA, A. K.; BHATIA, T. A review on social spam detection: challenges, open issues, and future directions. *Expert Systems with Applications*, v. 186, 115742, 2021. DOI: 10.1016/j.eswa.2021.115742.
RIZZO, Igor Vilela; CANATO, Robson Leandro Carvalho. Inteligência artificial: funções de ativação. *Prospectus*, v. 2, n. 2, 2020. ISSN 2674-8576.
ROSENBLATT, F.; PAPERT, S. *Perceptron*. v. 9, abr. 2021.
SAHAMI, M.; DUMAIS, S.; HECKERMAN, D.; HORVITZ, E. A Bayesian approach to filtering junk e-mail. *Machine Learning*, v. 43, n. 1, p. 98–112, 2021.
SAMARTHRAO, K. V.; ROHOKALE, V. M. A hybrid meta-heuristic-based multi-objective feature selection with adaptive capsule network for automated email spam detection. *International Journal of Intelligent Robotics and Applications*, v. 6, n. 3, p. 497-521, 2022. DOI: 10.1007/s41315-021-00217-9.
VASWANI, A. *et al*. Attention is all you need. *Advances in Neural Information Processing Systems*, v. 30, 2017. DOI: 10.48550/arXiv.1706.03762.
ZAKI, M. J.; MEIRA JR., W. *Data mining and machine learning: fundamental concepts and algorithms*. Cambridge: Cambridge University Press, 2020.
WEI, F.; NGUYEN, T. A lightweight deep neural model for SMS spam detection. In: *INTERNATIONAL SYMPOSIUM ON NETWORKS, COMPUTERS AND COMMUNICATIONS (ISNCC)*, 2020, [S. l.]. Anais [...]. Piscataway: IEEE, 2020. p. 1-6. DOI: 10.1109/ISNCC49221.2020.9297350.