INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS DISTRITO INDUSTRIAL
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

JOÃO PAULO SANTA RITA NEVES

IMPLEMENTAÇÃO DO MÉTODO DE REGRESSÃO LOGÍSTICA NA CLASSIFICAÇÃO DE EXAMES POR ESPECTROMETRIA DE MASSA QUANTO À PRESENÇA DE CÂNCER DO OVÁRIO.

MANAUS - AM

2021

JOÃO PAULO SANTA RITA NEVES

IMPLEMENTAÇÃO DO MÉTODO DE REGRESSÃO LOGÍSTICA NA CLASSIFICAÇÃO DE EXAMES POR ESPECTROMETRIA DE MASSA QUANTO À PRESENÇA DE CÂNCER DO OVÁRIO.

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Controle e Automação, do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Campus Manaus Distrito Industrial – IFAM/CMDI.

Orientador: Prof. Dr. Alyson de Jesus dos Santos

MANAUS - AM

2021

À minha Família.

AGRADECIMENTOS

Palavras não podem descrever o sentimento de gratidão perante a conclusão de mais uma jornada. Agradeço a minha família por todo o apoio e suporte em busca de meus sonhos e objetivos, sem eles nada seria possível, encontro em cada um, uma forma de inspiração.

Agradeço a Deus, pelo dom da vida, pela paciência e força me dada perante os momentos de aflição e angústia, a Ele entrego todos meus sonhos e propósitos, Ele me guia para todo o sempre.

Aos meus amigos de curso que somaram muito na minha formação, amigos esses que viraram irmãos.

Agradeço ao meu orientador, Prof. Dr. Alyson de Jesus dos Santos, pela sabedoria e paciência com que me orientou remotamente devido a situação de pandemia durante o ano de 2021, e por sempre estar disposto a me ajudar.

A Secretaria do Curso, pela cooperação.

Enfim, a todos os que de alguma forma contribuíram para a realização desta pesquisa e fim de mais esse ciclo.

‘‘Não temas, crê somente”.
(Marcos 5:36)

RESUMO

Este trabalho tem como objetivo implementar o método de regressão logística, uma rede neural simples de 01 única camada, para classificar os resultados dos exames de espectrometria de massa em 02 classes de diagnostico: com ou sem câncer de ovário. Foi utilizada uma base de dados "*ovarianInputs*" com os dados de 216 pacientes examinados com intensidades de íons correspondentes a 100 valores específicos de carga-massa, assim como a base de dados "*ovarianTargets*" com os resultados do diagnostico para fins de treinamento da rede neural (aprendizado supervisionado). Foi utilizado o método k-fold em 5 pastas randomizadas para a avaliação da acurácia média do modelo. Utilizou-se a matriz de confusão obtida a partir da classificação dos elementos do conjunto de teste de cada pasta. O algoritmo responsável por essa implementação foi desenvolvido utilizando as bibliotecas da linguagem Python e os resultados foram comparados com os resultados obtidos a partir da formulação matemática do modelo no software MATLAB, alcançando uma acurácia média de 93,03% em ambas as implementações.

Palavras-chave: Regressão Logística. Câncer de Ovário. Rede Neural.

ABSTRACT

This work aims to implement the logistic regression method, a simple 01 single-layer Artificial Neural Network, to classify the results of mass spectrometry exams in 02 diagnosis classes: with or without ovarian cancer. An “OvarianInputs” database was used with data from 216 patients examined with ion intensities corresponding to 100 specific load-mass values and the “OvarianTargets” database with diagnostic results for network training purposes neural. The k-fold cross validation was used in 5 randomized folders to assess the average accuracy of the model. The confusion matrix obtained from the classification of the elements of the test set of each folder was used. The algorithm responsible for this implementation was developed using Python language libraries and compared with the results obtained from the mathematical formulation of the model in MATLAB software, reaching an average accuracy of 93.03% in both implementations.

Keywords: Logistic Regression. Ovary Cancer. Artificial Neural Network.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

$FCecon$	Fundação Centro de Controle de Oncologia do Amazonas
$SES-AM$	Secretaria de Estado de Saúde
SUS	Sistema Único de Saúde
$BRCA$	Breast Cancer
EM	Espectrometria de Massa
$MALDI$	Matrix Assistes Laser Desorption Ionization
ESI	Electron Spray Ionization
RNA	Ácido Desoxoribonucleico
$Na+$	Sódio
$K+$	Potássio
$Cl-$	Cloreto
x_j	Entrada Neurônio Artificial
j	Sinapse
k	Neurônio
w_{kj}	Peso Sináptico
$\pi(x)$	Função Monotônica
x	Variável explicativa
Y	Variável resposta
$f.d.a$	Função de Distribuição Acumulada
TFN	Taxa de erro falso negativo
TFP	Taxa de erro falso positivo
TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
ROC	*Receiver Operating Characteristic*

SUMÁRIO

1 INTRODUÇÃO

Segundo o Instituto Nacional de Câncer (INCA), o câncer de ovário é considerado o câncer ginecológico mais difícil de ser diagnosticado, pois a maioria dos tumores malignos de ovário são descobertos tardiamente, quando já estão em estágio avançado. E, consequentemente, faz com que o câncer de ovário seja o câncer ginecológico mais letal. Este é um dos fatores que instiga estudos sobre maneiras de predizê-los a partir de exames médicos específicos aliados a modelos de *Machine Learning*.

O Instituto Nacional de Câncer (INCA) estima que, no Brasil, para cada ano do triênio 2020/2022, sejam diagnosticados no Brasil 6.650 novos casos de câncer de ovário, com um risco estimado de 6,18 casos a cada 100 mil mulheres. O câncer de ovário ocupa o quinto lugar em mortes por câncer entre as mulheres, sendo responsável por mais mortes do que qualquer outro câncer do sistema reprodutivo feminino. O risco de uma mulher desenvolver câncer de ovário durante sua vida é de 1 em 78. A chance de uma mulher morrer de câncer de ovário é de cerca de 1 em 108. Essas estatísticas não levam em conta tumores ovarianos de baixo potencial de malignidade (INCA 2020).

No Amazonas, segundo a Fundação Centro de Controle de Oncologia do Estado do Amazonas (FCecon), unidade vinculada à Secretaria de Estado de Saúde (SES-AM), o câncer de ovário deve atingir pelo menos 80 mulheres em 2021.

A regressão logística consiste em uma técnica de mineração de dados que vem sendo aplicada intensamente em várias áreas de conhecimento e, em especial, na área médica, pois o estudo de variáveis respostas binarias em função de um conjunto de fatores explicativos vem se tornando comumente usados em trabalhos relacionados a área da saúde.

Segundo Hosmer e Lemeshow (2000), os métodos de regressão vêm se tornando um componente fundamental para as análises de dados interessadas em descrever a relação entre uma variável resposta (dependente) e uma ou mais variáveis explicativas (independentes).

O modelo de regressão logística é semelhante ao modelo de regressão linear, entretanto, a principal diferença, é que a variável dependente na regressão logística é categórica, e segundo Agresti (2002), o modelo logístico é o mais importante para dados de resposta categórica. Além disso, a variável dependente frequentemente é

binaria, assim, assumindo dois valores, que geralmente são tratados como “sucesso” ou “fracasso”.

Consoante a Mesquita (2014), embora a regressão logística seja aplicada mais frequentemente na área médica, essa técnica tem mostrado grande eficiência nas mais diversas áreas de estudo, desde ciências médicas, a estudos de mercado. Na área médica em especial, sua usabilidade é geralmente relacionada a análises e predições sobre assuntos cancerígenos.

Diante das questões apresentadas, no presente estudo, foi implementado o modelo de Regressão Logística de 01 camada e aplicado à base de dados *’OvarianInputs’*, com os resultados de exames de espectrometria de 216 pacientes. Tais exames mediram as intensidades de íons correspondentes a 100 valores específicos de carga-massa a fim de determinar o diagnóstico dos pacientes com ou sem câncer de ovário. Esse é um caso de aprendizado supervisionado, uma vez que os resultados (0:sem câncer, 1: com câncer), já conhecidos para este conjunto de dados de exames, serão utilizados para a obtenção dos pesos ou coeficientes que descrevem a equação do modelo de classificação.

1.1 OBJETIVOS

1.2.1 Objetivo Geral

A partir de um conjunto de dados obtidos em *databases* fornecidas pelo Centro para Pesquisa do Câncer do Instituto Nacional do Câncer, contendo *resultados* de 216 pacientes sobre dados de exames de espectrometria de massa, implementar e avaliar um modelo baseado em métodos de aprendizado de máquina não supervisionado para classificar os dados em duas classes. Em seguida, os casos com melhores resultados de predição serão mostrados e explanados.

1.2.2 Objetivo Específico

- Realizar análise exploratória baseada em métodos supervisionados de aprendizado de máquina;
- Implementar e avaliar um modelo;

- Implementar o método de Regressão Logística com uma camada;
- Utilizar validação cruzada *K-fold* para treinamento e teste do modelo;
- Avaliar o desempenho dos algoritmos propostos;

1.2 JUSTIFICATIVA

A regressão logística tem uma vasta aplicação em diversas áreas do conhecimento. Na área médica é preciso reconhecer a doença (variável de efeito, o desfecho) e quais fatores estão envolvidos no seu aparecimento e na sua evolução (variáveis preditivas, fatores de exposição ou simplesmente exposição). Esses fatores podem representar associação entre exposição e doença, sendo assim, fatores de risco ou de proteção.

O câncer de ovário, por sua vez, é o segundo tumor ginecológico mais comum no Brasil, ficando atrás apenas do cervical. Isso torna ainda mais preocupante o resultado do estudo realizado pelo Observatório de Oncologia, que mostra que 68% das brasileiras atendidas pelo Sistema Único de Saúde (SUS) são diagnosticadas quando a doença já está em estágio avançado (Oncoguia, 2021).

As principais contribuições desse projeto de TCC podem ser sumarizadas em:

- Conscientização ao Câncer de Ovário;
- Uso de Técnicas de Aprendizado de Máquina no auxílio para detecção do câncer;
- Destacar a importância das Redes Neurais aliadas à área médica;

1.3 METODOLOGIA

A partir das bases de dados escolhidas, serão extraídos os datasets e classificados de acordo com o método de Regressão escolhido. Após isso, será feita uma avaliação do desempenho do algoritmo proposto

1.4 ESTRUTURA DO TCC

O restante deste TCC está estruturado da seguinte maneira. No Capítulo 2, é apresentada a fundamentação teórica com o propósito de mostrar uma explicação sobre o câncer de ovário e a técnica de espectrometria de massa, aprendizado de máquina e a técnica de regressão proposta e as métricas de avaliação a serem utilizadas. O Capítulo 3, apresenta os métodos e materiais utilizados a fim de mostrar as ferramentas computacionais utilizadas, descrição e implementação do modelo proposto. No Capítulo 4 é apresentado os resultados e discussões. No Capítulo 5 é apresentada a conclusão do trabalho. Por fim, no Capítulo 6, apresentamos os trabalhos futuros propostos.

2 REFERENCIAL TEÓRICO

Neste capítulo, é apresentado um embasamento teórico referente a câncer de ovário, aprendizado de máquina e redes neurais artificiais. Em seguida, é realizado um estudo sobre a técnica de regressão logística. Posteriormente, são apresentados os conceitos das métricas de avaliação utilizadas. Por fim, apresenta-se as considerações finais deste capítulo.

2.1 BREVE EXPLICAÇÃO SOBRE O CÂNCER DE OVÁRIO

O câncer de ovário é uma doença complexa, de difícil detecção nos estágios iniciais e de difícil tratamento. Como o ovário possui diversos tipos de células em sua superfície e seu interior, cada tipo celular apresenta comportamentos, tratamentos e prognósticos diferentes. A maioria dos tumores de ovário são carcinomas epiteliais, isto é, com origem nas células da superfície do ovário. Nas fases iniciais raramente apresentam sintomas e quando isto ocorre em geral são sintomas vagos (Oncomed, 2020).

Pacientes que apresentam história familiar de câncer de mama, ovário ou são portadoras de mutação do gene BRCA (gene que afeta a chance de a pessoa desenvolver câncer de ovário) têm maior risco de desenvolver câncer de ovário e devem ficar atentas a qualquer sintoma. Infelizmente, até o momento não existem exames preventivos para o câncer de ovário e o diagnóstico em fases iniciais é muito raro.

O diagnóstico é feito com base em exames radiológicos (tomografia, ressonância, ultrassonografia com doppler) e exames de sangue chamados marcadores tumorais. Cabe ressaltar que os marcadores tumorais podem estar alterados em várias outras doenças benignas. A única maneira de se confirmar o diagnóstico de um câncer de ovário é através da cirurgia, onde será realizada biópsia para correta identificação do tipo celular e avaliar a extensão da doença (cirurgia de estadiamento). Nos casos iniciais, onde a doença é diagnosticada após a retirada de um cisto ovariano, é necessária nova intervenção cirúrgica para retirada do útero e do outro ovário. Nos estágios avançados pode ser necessário a retirada de parte de outros órgãos acometidos pelo tumor.

Em casos em que a cirurgia proposta é demasiadamente extensa ou em pacientes muito debilitadas pode-se optar por iniciar o tratamento com quimioterapia, sendo indicada cirurgia posteriormente. Em situações excepcionais, onde a paciente deseja preservar a fertilidade, pode-se realizar uma cirurgia conservadora com preservação do útero e do outro ovário. Porém, esta conduta vai depender do estadiamento da doença e do tipo histológico (célula) do tumor.

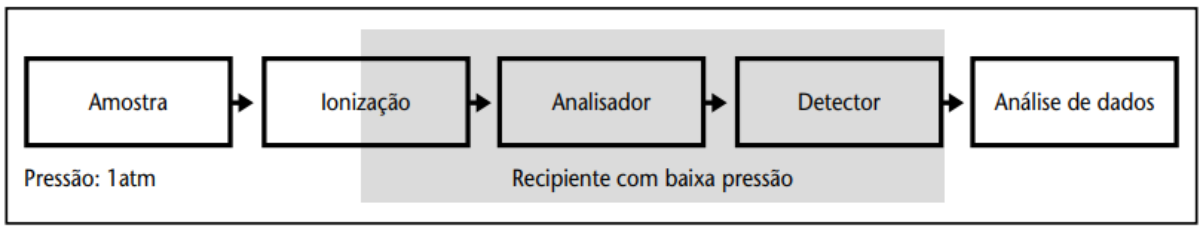
O prognóstico do câncer de ovário em estágios iniciais é razoável, sendo que nos casos avançados é comum ocorrer retorno do tumor. Nestes casos a opção por novo tratamento com cirurgia ou quimioterapia vai depender da extensão do novo tumor e da resposta à quimioterapia.

Após o tratamento a paciente deverá ser acompanhada periodicamente, sendo realizada consulta clínica, exame ginecológico, solicitação de exames de imagem e marcadores tumorais.

2.1.1 EXAME DE ESPECTROMETRIA DE MASSA

O espectrômetro de massa é um instrumento analítico capaz de converter moléculas neutras em íons na forma gasosa e separá-las de acordo com a sua razão massa/carga (m/z), utilizando para isso campos eletromagnéticos. Esse equipamento atua como uma balança de íons de altíssima precisão e, em sua grande maioria, é composto por uma fonte ionizante, analisador(es), e detector(es), conforme esquematizado na Figura 1.

Figura 1 - Esquema simplificado de um espectrômetro de massa



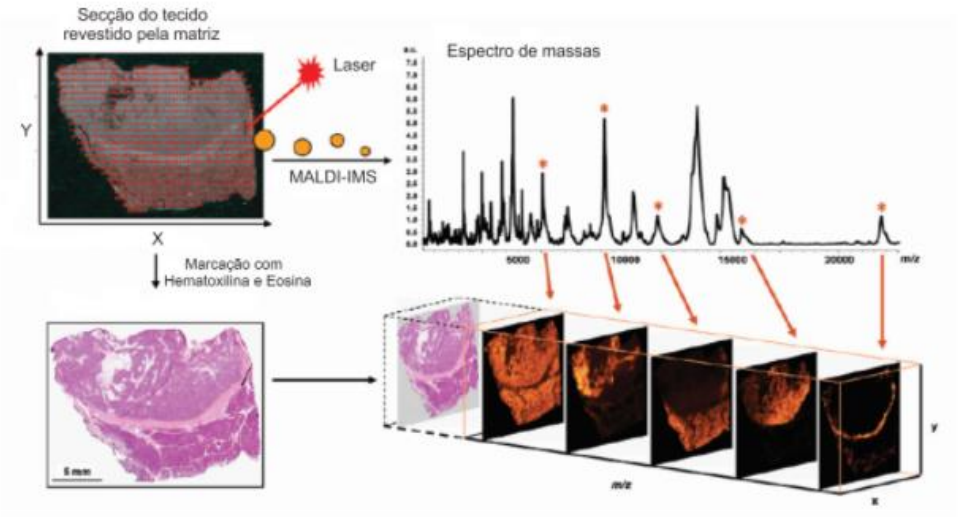
Fonte: CARVALHO et al. (2006)

Apesar de a Espectrometria de Massa (EM) ser uma tecnologia usada há bastante tempo no campo das engenharias e da física, era pouco utilizada nas

ciências da saúde por degradar biomoléculas ao ionizá-las. Recentemente foram desenvolvidas duas técnicas capazes de ionizar biomoléculas de alto peso molecular, como oligo peptídeos e proteínas (Vestal, 2001). Estas técnicas também são capazes de alterar o analito de sua fase sólida ou líquida para gasosa, estado necessário para realização da análise da EM sem degradá-lo. Tais técnicas foram denominadas *matrix assisted laser desorption ionization* (MALDI) e *electron spray ionization* (ESI) aperfeiçoada para estudos de macromoléculas biológicas (Fenn et al., 1989).

A Figura 2 mostra o fluxo de trabalho de uma técnica de espectrometria de massa para detecção de câncer de ovário, onde uma secção do tecido predisposto ao câncer é analisada.

Figura 2 - Espectrometria para diagnóstico do Câncer de Ovário.



Fonte: Carvalho et al. (2013)

2.2 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina, ou *Machine Learnig,* pode ser entendido como uma metodologia para inteligência artificial, em que os computadores são ensinados a realizarem determinadas tarefas sem uma programação prévia explícita. É baseada principalmente em observações e experiências do mundo real (que são convertidas

em dados) e servem como entrada para o aprendizado de máquina. Dessa forma, pode ser considerada uma forma de programação por amostra de dados (Khan, 2018).

O aprendizado de máquina na área médica está se tornando mais amplamente usado e está ajudando pacientes e profissionais de muitas maneiras diferentes. Os casos de uso na saúde mais comuns são a automação do faturamento médico, o suporte à decisão clínica, desenvolvimento de diretrizes de atendimento clínico e aplicações ligadas a detecção de câncer. O aprendizado de máquina é realizado de três formas: supervisionada, não supervisionada e de reforço.

O foco deste trabalho é voltado ao aprendizado supervisionado. No aprendizado supervisionado é fornecida uma referência do objetivo a ser alcançado, isto é, um treinamento com o conhecimento do ambiente. Estes treinamentos são conjuntos de exemplos com entradas e uma saída esperada. O algoritmo de aprendizado de máquina extrai a representação do conhecimento a partir desses exemplos. O objetivo é que a representação gerada seja capaz de produzir saídas corretas para novas entradas não apresentadas antes.

2.3 REDES NEURAIS ARTIFICIAIS (RNA'S)

Desde as primeiras concepções sobre redes neurais artificiais, datadas de 1943, por Warren McCulloch e Walter Pitts (Fleck, 2016), os estudos em neuro computação foram avançando de tal forma a permitirem grandes avanços, que hoje possibilitam a resolução de problemas complexos como a detecção de objetos em tempo real por meio de aprendizagem profunda.

Segundo (Haykin, 1998), uma rede neural é um processador maciçamente e paralelamente distribuído, constituído de unidades de processamento simples, que têm a propensão natural de armazenar conhecimento experimental e torná-lo disponível para o uso.

Numa abordagem mais detalhada, RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples (nodos) que calculam determinadas funções matemáticas (normalmente não-lineares). Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos estas conexões estão

associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede. O funcionamento destas redes é inspirado em uma estrutura física concebida pela natureza: o cérebro humano (Braga e Carvalho, 2000).

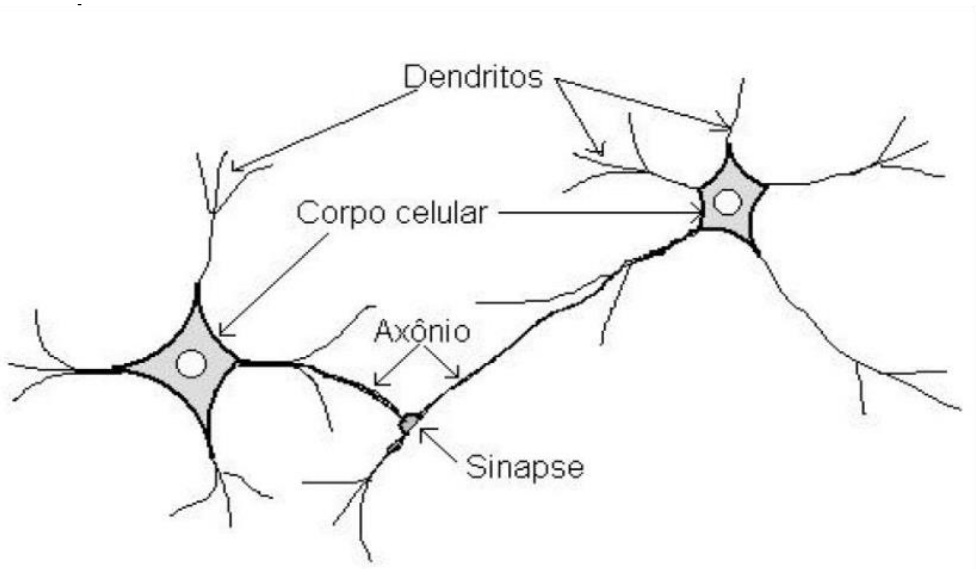
2.3.1. NEURÔNIO BIOLÓGICO

O neurônio é um dos tipos de célula presentes no cérebro humano, e são extremamente numerosos (aproximadamente 1011 unidades). A Figura 3 mostra uma representação simplificada de um neurônio.

O corpo celular, ou soma, possui organelas as quais podem ser encontradas na maioria das células do corpo humano, como por exemplo núcleo, mitocôndrias e complexo de Golgi. Uma característica morfológica que diferencia as células nervosas das demais é a presença de fibras que se ramificam do corpo celular. Uma dessas fibras, o axônio (que geralmente é única), é a responsável por transmitir sinais para outros neurônios, e pode ser considerado como saída do neurônio. O restante das fibras é chamado de dendritos, que tem a função de transmitir os sinais provenientes de outros neurônios para o corpo celular, portanto se comportando como entradas neurais.

De forma geral, cada neurônio possui vários dendritos e apenas um axônio. A extremidade do axônio é conectada com dendritos de outros neurônios pelas sinapses, que é uma região entre as células em que são estabelecidas as comunicações (Gurney, 1997; Barreto, 2002).

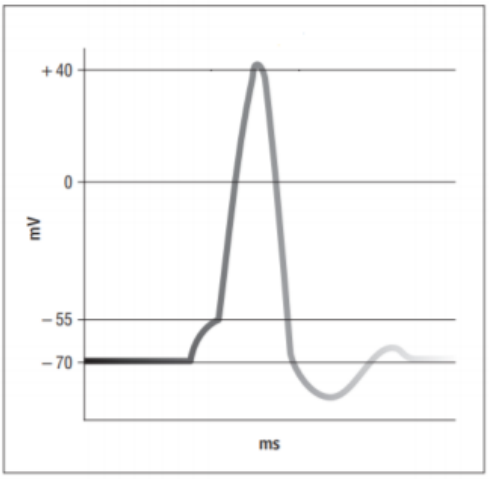
Figura 3 - Neurônio biológico típico



Fonte: (BARRETO, 2002)

As informações que são transmitidas de um neurônio a outro acontecem por meio de impulsos elétricos, denominados potenciais de ação. Pelo fato de a membrana celular ser semipermeável, apenas algumas substâncias podem passar entre ela, como por exemplo o Na+, k+ e Cl−. As concentrações dessas substâncias nos meios intracelular e extracelular são diferentes, o que leva a uma diferença de potencial entre esses meios, sendo que no repouso, o valor é de aproximadamente -70mV. Para que haja a deflagração do potencial de ação, um impulso sináptico (gerado por exemplo por um mecanismo de ativação mecânico, como uma vibração sonora) deve elevar esse valor a aproximadamente -55mV, e então o neurônio dispara um sinal cujo valor da diferença de potencial atinge um pico de cerca de 40mV, e alguns milissegundos depois o estado de repouso é reestabelecido, conforme mostra a Figura 4 (Krueger-Beck, 2011).

Figura 4 - Característica do potencial de ação



Fonte: Adaptado de (KRUEGER-BECK, 2011)

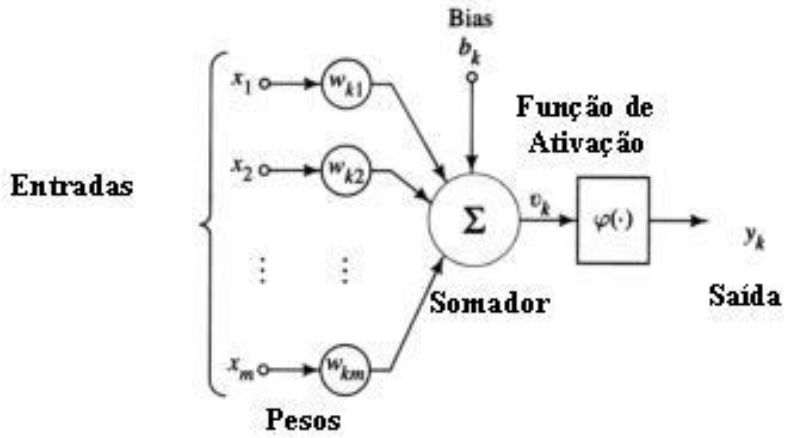
2.3.2. NEURÔNIO ARTIFICIAL

É possível modelar matematicamente um neurônio artificial por meio dos conceitos de um neurônio biológico, e com isso formar a peça básica para a construção de RNAs. Para tanto, são identificados três elementos básicos de um modelo de neurônio (Haykin, 1998):

- Um conjunto de sinapses, que são caracterizados por um peso. Mais especificamente, uma entrada x_j de uma sinapse j conectada ao neurônio k é multiplicada pelo peso sináptico w_{kj}.
- Um somador para somar os sinais de entradas, ponderadas pelos respectivos pesos sinápticos do neurônio. Essa operação configura uma combinação linear.
- Uma função de ativação que limita a amplitude do sinal de saída de um neurônio.

A Figura 5 mostra um modelo de neurônio artificial.

Figura 5 - Modelo de neurônio artificial.



Fonte: Adaptado de (HAYKIN, 1998)

2.4 REGRESSÃO LOGÍSTICA

A regressão logística é uma forma de modelagem estatística que é frequentemente utilizada em situações em que a variável resposta é de natureza dicotômica. Isso exige que o resultado da análise possibilite associações a certas categorias, descrevendo a relação entre a variável resposta categórica e um conjunto de variáveis explicativas, nas quais podem ser categóricas ou métricas. De acordo com Figueira (2006), a variável resposta é geralmente dicotômica, mas pode ser politômica, isto é, tem mais do que dois níveis de resposta. As categorias (ou valores) que a variável dependente assume podem ser de natureza nominal ou ordinal. Na situação de natureza ordinal, há uma ordem natural entre as possíveis categorias e, então, se tem o contexto da Regressão Logística Ordinal. Quando não existe esta ordem entre as categorias da variável dependente, tem-se o contexto da Regressão Logística Nominal.

O modelo de regressão logística é uma extensão da análise de tabelas de múltipla entrada para a estrutura de análise de regressão, na qual se modelam os resultados de probabilidades binomiais, além de que é um caso particular dos modelos lineares generalizados com componente aleatório binomial e função de ligação logit.

2.4.1. REGRESSÃO LOGÍSTICA SIMPLES

Inicialmente considera-se que $\pi(x)$ é uma função monotônica com valores entre zero e um, quando x varia na reta real, ou seja, $\pi(x)$ é uma função de distribuição de probabilidade, na qual representa a probabilidade de "sucesso", dado o valor de x de uma variável explicativa qualquer. A variável resposta Y é dicotômica, assumindo o valor 1 para o evento de interesse (sucesso) e o valor 0 para evento complementar (fracasso).

Considera-se uma série de eventos binários, onde $(Y_1, Y_2, \dots, Y_n)$ são variáveis aleatórias independentes com distribuição Bernoulli, com probabilidade de sucesso $\pi(x)$, isto é, $Y_1 \sim Ber(\pi(x))$, consoante a Souza (2006). Desta forma, $\pi(x) = P(Y = 1|X = x) = 1 - P(Y = 0|X = x)$ e $0 < \pi(x) < 1$. Como $\pi(x)$ varia entre zero e um, uma representação linear para ela sobre todos os valores de x não é adequada, então se considera a transformação logística de $\pi(x)$ sob a forma linear:

$$\ln\left[\frac{\boldsymbol{\pi(x)}}{\mathbf{1-\pi(x)}}\right] = \boldsymbol{logit}[\boldsymbol{\pi(x)}] = \boldsymbol{\beta_0 + \beta_1 x}, \tag{1}$$

ou equivalente

$$\boldsymbol{\pi(x)} = \frac{\mathbf{exp}\,(\boldsymbol{\beta_0 + \beta_1 x})}{\mathbf{1 + exp}\,(\boldsymbol{\beta_0 + \beta_1 x})} \tag{2}$$

sendo $\boldsymbol{\beta_0}$ e $\boldsymbol{\beta_1}$ parâmetros desconhecidos. Conforme $\boldsymbol{x \to \infty}$, $\boldsymbol{\pi(x) \downarrow 0}$ quando $\boldsymbol{\beta_1 < 0}$, e $\boldsymbol{\pi(x) \uparrow 1}$ quando $\boldsymbol{\beta_1 > 0}$. Enquanto $\boldsymbol{\pi(x)}$ deve cair no intervalo $(\mathbf{0}, \mathbf{1})$, o $\boldsymbol{logit}[\boldsymbol{\pi(x)}]$ pode ser qualquer número real (AGRESTI, 2002).

Conforme Souza (2006), em qualquer problema de regressão a quantidade a ser modelada é a esperança da variável aleatória dependente, dado o valor da variável independente, ou seja, $\mathbb{E}(Y|X = x)$. Devido à natureza da variável resposta, $0 \leq \mathbb{E}(Y|X = x) \leq 1$ na regressão logística, enquanto na regressão linear $-\infty \leq \mathbb{E}(Y|X = x) \leq \infty$. Na regressão linear, $\mathbb{E}(Y|X = x) = \beta_0 + \beta_1 x$ e na regressão logística, usando a definição de variáveis aleatórias discretas, tem-se que

$$\mathbb{E}(Y|X = x) = 1P(Y = 1|X = x) + 0P(Y = 0|X = x) = \pi(x).$$

Outra diferença importante entre o modelo de regressão linear e o modelo de regressão logístico refere-se à distribuição condicional de Y. No modelo linear assume-se que uma observação possa ser expressa como $y = \mathbb{E}(Y|x) + \epsilon$, com a suposição de que o erro ϵ tenha distribuição Normal com média zero e variância constante. Este não é o caso quando Y é dicotômica. Desta maneira, uma observação pode ser expressa como $y = \pi(x) + \epsilon$, em que ϵ pode assumir uma de duas possibilidades: se $y = 1$, então $\epsilon = 1 - \pi(x)$ com probabilidade $\pi(x)$, e se $y = 0$, com $\epsilon = -\pi(x)$ probabilidade $1 - \pi(x)$. Portanto, ϵ tem distribuição Bernoulli com média zero e variância $\pi(x)$[1- $\pi(x)$].

Para cada x_i para $i \epsilon \{1, \dots, n\}$, segue-se que $\mathbb{E}(Y_i|x_i) = \pi(x_i)$. Cada observação y_i pode ser interpretada, para cada $i \epsilon \{1, \dots, n\}$, como

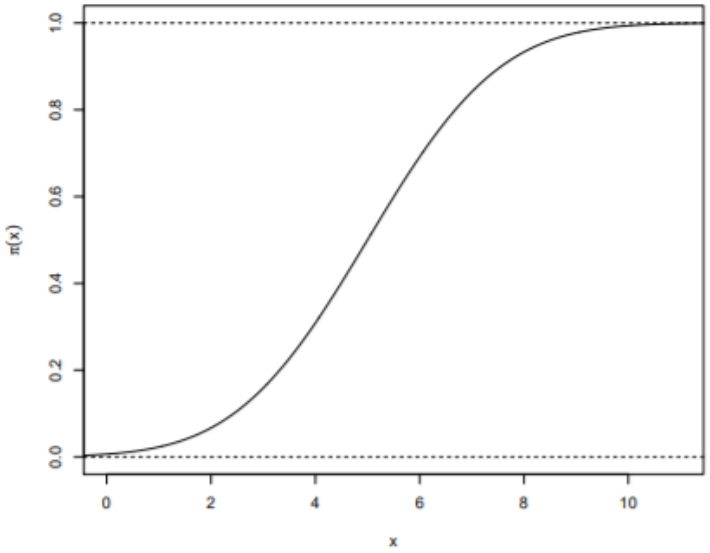
$$y_i = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)} + \epsilon_i$$

onde os erros ϵ_i seguem as seguintes suposições, para todo $i, l \epsilon \{1, \dots, n\}$ (FIGUEIRA, 2006):

$$(i)\ \mathbb{E}(\epsilon_i|x_i) = 0$$
$$(ii)\ Var(\epsilon_i|x_i) = \pi(x_i)[1 - \pi(x_i)]$$
$$(iii)\ Cov(\epsilon_i, \epsilon_l) = 0, se\ i \neq l$$

Devido à estimação de probabilidades estar compreendida nos limites (0,1), é de se esperar que as mudanças ocorridas na variável independente produzam efeitos cada vez menores sobre a variável resposta à medida que ela assume valores mais próximos dos extremos. A curva de regressão monótona apresentada na Figura 6 tem o formato de uma função distribuição acumulada (f.d.a) para uma variável aleatória contínua quando $\beta_1 > 0$. Isso sugere um modelo para resposta binária tendo a forma $\pi(x) = F_X(x)$ de alguma f.d.a F_X.

Figura 6 – Curva Logística ou Sigmoide



Fonte: Autoria Própria (2021)

Quando $\beta_1 > 0$, a curva de regressão logística é uma função de distribuição acumulada da distribuição logística. A f.d.a de uma distribuição logística com média μ e parâmetro de dispersão $\tau > 0$ é da seguinte forma:

$$F_X(x) = \frac{exp[(x - \mu)/\tau]}{1 + exp[(x - \mu)/\tau]}, -\infty < x < \infty.$$

A forma padronizada da f.d.a logística tem $\mu = 0$ e $\tau = 1$. Seja $\Phi(\cdot)$ denotando uma f.d.a padrão, então $\Phi(x) = e^x/(1 + e^x)$. Para esta função, a curva de regressão logística tem a forma $\pi(x) = \Phi(\beta_0 + \beta_1 x)$. A transformação *logit* é simplesmente a função inversa da função distribuição acumulada logística padrão; isto é, quando $\Phi(x) = \pi(x) = e^x/(1 + e^x)$, então $x = \Phi^{-1}[\pi(x)] = ln[\pi(x)/(1 - \pi(x))]$ (AGRESTI, 2002).

2.4.1.1. ESTIMAÇÃO DE COEFICIENTES

Identificada a equação que permite calcular a probabilidade relativa à ocorrência de determinado evento, resta estimar os seus coeficientes. Devido à variância não constante, o estimador de máxima verossimilhança é mais eficiente do que o estimador de mínimos quadrados.

Supõe-se uma amostra de n observações independentes de pares (x_i, y_i) para $i \in \{1, \ldots, n\}$, em que y_i representa o valor da variável aleatória Y e x_i o valor da variável independente para a i-ésima observação. Seja $\beta = (\beta_0, \beta_1)$ o vetor de parâmetros relacionado com a probabilidade condicional $P(Y_i = 1|x_i) = \pi(x_i)$. Conforme Figueira (2006), a função de verossimilhança para o modelo logístico é dada por

$$L(\beta) = \prod_{i=1}^{n} \pi(x_i)^{y^i} \, [1 - \pi(x_i)]^{(1-y_i)}. \quad (3)$$

2.4.1.2. INTERPRETAÇÃO DOS COEFICIENTES

Paula (2010) diz que o modelo apresentado na equação (1) poderia, por exemplo, ser aplicado para analisar a associação entre uma determinada doença e a ocorrência ou não de um fator particular. Seriam então amostrados, independentemente, n_1 indivíduos com presença do fator $(x = 1)$ e n_2 indivíduos com ausência do fator $(x = 0)$ e seria a probabilidade de desenvolvimento da doença após certo período fixo. Dessa forma, a chance de desenvolvimento da doença para um indivíduo com presença do fator fica dada por

$$\frac{\pi(1)}{1 - \pi(1)} = e^{\widehat{\beta_0} + \widehat{\beta_x}}$$

enquanto a chance de desenvolvimento da doença para um indivíduo com ausência do fator é simplesmente

$$\frac{\pi(0)}{1 - \pi(0)} = e^{\widehat{\beta_0}}$$

2.4.2. REGRESSÃO LOGÍSTICA MÚLTIPLA

Aqui será generalizado o modelo logístico para o caso de mais de uma variável independente, ou seja, o caso múltiplo. Considera-se um conjunto de p variáveis independentes denotadas por $\boldsymbol{X} = (X_1, \ldots, X_P)^T$, em que $\boldsymbol{x} = (x_1, \ldots, x_p)^T$ ´e um valor particular e uma v.a dependente binária Y . Denotando por $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ o vetor de parâmetros desconhecidos e β_j sendo o j-ésimo parâmetro associado à variável explicativa x_j , com $j = 0, 1, \ldots, p$, então a transformação logística de $\pi(x)$ sob a forma linear é dada por

$$\ln\left[\frac{\pi(\boldsymbol{x})}{1-\pi(\boldsymbol{x})}\right] = logit[\pi(\boldsymbol{x})] = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p. \quad (4)$$

Então, o modelo para predizer $P(Y = 1|\boldsymbol{x}\_ = \pi(\boldsymbol{x})$ é

$$\pi(\boldsymbol{x}) = \frac{\exp(\beta_0+\beta_1 x_1+\cdots+\beta_p x_p)}{1+\exp(\beta_0+\beta_1 x_1+\cdots+\beta_p x_p)} \quad (5)$$

Conforme Figueira (2006), dadas n observações independentes de Y , denotadas por $(y_1, \ldots, y_n)$, associadas aos valores de $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$ para $i \in \{1, \ldots, n\}$, o logit dado pela equação (4) apresenta-se da forma

$$logit[\pi(\boldsymbol{x}_i)] = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \in_i$$

onde os erros $\in_i$ seguem as seguintes suposições, para todo $i, l \in \{1, \ldots, n\}$:

$$(i)\ \mathbb{E}(\epsilon_i|\boldsymbol{x_i}) = 0$$
$$(ii)\ Var(\epsilon_i|\boldsymbol{x_i}) = \pi(\boldsymbol{x_i})[1 - \pi(\boldsymbol{x_i})]$$
$$(iii)\ Cov(\epsilon_i, \epsilon_l) = 0, se\ i \neq l \quad (6)$$

Assim, as v.a's $Y_1, \ldots, Y_n$satisfazem o lodelo logístico múltiplo se uma amostra de tamanho um de cada Y_i pode ser expressa como

$$y_i = \frac{\exp\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right)}{1 + \exp\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right)} + \epsilon_i$$

para qual x_{ij} é constante conhecida, β_j é parâmetro desconhecido do modelo e os erros ϵ_i possuem as suposições dadas em (6).

2.4.2.1 ESTIMAÇÃO DOS COEFICIENTES

Para a estimação dos parâmetros foi utilizado o método de máxima verossimilhança similar ao caso do modelo logístico simples. No caso do modelo múltiplo, a função de verossimilhança é dada pela expressão (5). Seja β o vetor de parâmetros relacionado com a probabilidade condicional $P(Y_i = 1|\boldsymbol{x}_i) = \pi(\boldsymbol{x}_i)$. para $i \in \{1, \ldots, n\}$. Então, para uma amostra de tamanho n, tem-se que

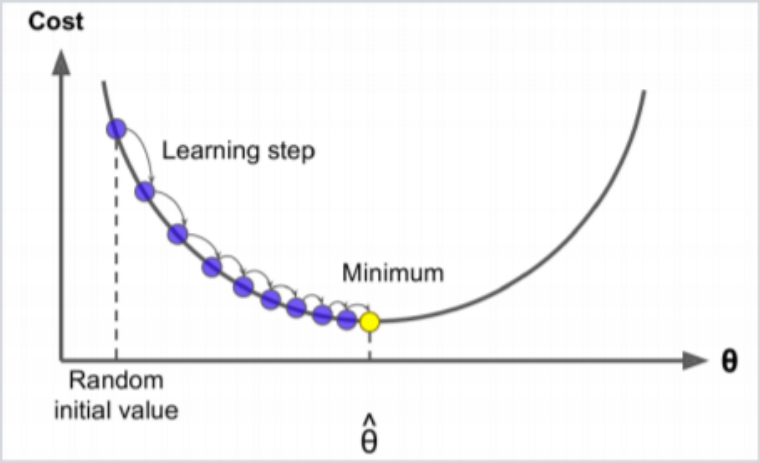
$$L(\beta) = \prod_{i=1}^{n} \pi(\boldsymbol{x}_i)^{y^i} \ [1 - \pi(\boldsymbol{x}_i)]^{(1-y_i)}, com\ (y_i \in \{0,1\} \qquad (7)$$

2.5 GRADIENTE DESCENDENTE

Gradiente descendente é um dos algoritmos de maior sucesso em problemas de *Machine Learning*. O método consiste em encontrar, de forma iterativa, os valores dos parâmetros que minimizam determinada função de interesse.

A ideia geral é através de processos de iteração de parâmetros, encontrar os valores que minimizem o erro quadrático entre a hipótese e a função alvo.

Figura 7 - Gradiente local da função erro em relação ao vetor de parâmetros θ



Fonte: https://ichi.pro/pt/conceito-de-gradiente-descendente-em-aprendizado-de-maquina-75821333878444 (2021)

A operação derivada consegue informar a intensidade (ou a sensibilidade) da variação de uma função em relação a suas variáveis independentes.

O modelo algébrico da variação função custo em relação aos parâmetros a serem otimizados é dado por,

$$\frac{\partial MSE(\theta)}{\partial \theta_j} = \frac{2}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})x_j^i \quad (8)$$

O modelo vetorizado,

$$\nabla_\theta MSE(\theta) = \frac{2}{m}x^T(x * \theta - y) \quad (9)$$

2.6 MÉTRICAS DE AVALIAÇÃO

As métricas de avaliação servem para rastrear um fator de risco de determinado experimento. Essas métricas, que serão descritas a seguir, auxiliam no ajuste do modelo de regressão logística.

2.6.1 RESULTADOS FALSO-POSITIVOS E FALSO-NEGATIVOS

Tomando como exemplo um estudo de avaliação de determinada doença, há o interesse de saber se as pessoas testadas têm ou não a doença. O resultado do

teste pode ser positivo (prevendo que a pessoa está doente) ou negativo (prevendo que a pessoa não está doente), podendo ou não coincidir com a verdadeira situação da pessoa. Desta forma, tem-se as seguintes definições:

- Falso-positivo: ocorre quando o teste é positivo, mas o indivíduo não está doente. E também conhecido como erro tipo I
- Falso-negativo: ocorre quando o teste é negativo, mas o indivíduo está doente. E também conhecido como erro tipo II.

2.6.2 SENSIBILIDADE E ESPECIFICIDADE

Sensibilidade e especificidade são duas medidas importantes do funcionamento de um teste. A sensibilidade (pode também ser chamada de taxa de verdadeiro positivo) refere-se à capacidade de um teste para detectar uma doença quando ela está presente, ou seja, é a probabilidade de um resultado positivo de um teste, dado que o indivíduo realmente esteja doente. O teste será altamente sensível se a probabilidade for alta e o teste não será sensível se ele falhar na detecção da doença em indivíduos doentes. A taxa com que isso ocorre é chamada de taxa de erro falso-negativo (TFN).

A especificidade (também chamada de taxa de verdadeiro negativo) é capacidade de um teste indicar ausência de doença quando ela não está presente, ou seja, é a probabilidade de um resultado negativo de um teste, dado que o indivíduo não está doente. Quando essa probabilidade é alta, o teste é altamente específico e, se o teste não é específico, indicará falsamente a presença de doença em indivíduos não-doentes. A taxa com que isso ocorre ´e chamada de taxa de erro falso-positivo (TFP). A especificidade e a taxa de erro falso-positivo somam 1 (100%).

O teste ideal, com 100% de sensibilidade e especificidade, raramente existe na prática, pois a tentativa de melhorar a sensibilidade frequentemente tem o efeito de diminuir a especificidade.

2.6.3 MATRIZ DE CONFUSÃO

Matriz de confusão é uma tabela que mostra as frequências de classificação para cada classe do modelo.

De acordo com (Godbole; Sarawagi, 2004), uma matriz de confusão para um problema de n classes e uma matriz M de ordem $n \times n$, no qual o elemento M_{ij} corresponde ao número de classificações corretas para a classe i, quando $i = j$. Para $i \neq j$, o elemento M_{ij} corresponde ao número de classificações da classe i que foram definidas incorretamente como classe j, e vice-versa.

Como estabelecido por (Douglas, 2019), o número de acertos para cada classe se localiza na diagonal principal da matriz M. Usualmente podem ser depreendidas algumas metricas de avaliação do modelo de classificação apenas com base na sua matriz de confusão. Neste trabalho, serão utilizadas as métricas de acurácia, sensibilidade e especificidade, definidas nas expressões (10), (11) e (12).

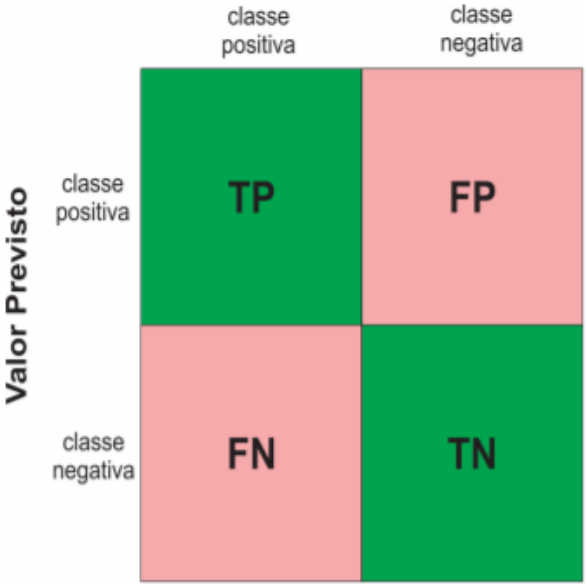
$$acurácia = \frac{acertos}{número\ de\ classificações} \tag{10}$$

$$sensibilidade = \frac{acertos\ de\ casos\ positivos}{número\ de\ casos\ positivos} \tag{11}$$

$$especificidade = \frac{acertos\ de\ casos\ negativos}{número\ de\ casos\ negativos} \tag{12}$$

A Figura 8 mostra o formato de uma matriz de confusão.

Figura 8 - Matriz de Confusão



Fonte: Autoria própria (2021)

2.7 VALIDAÇÃO CRUZADA

No Aprendizado de Máquina**,** a validação cruzada é um método de re-amostragem usado para avaliação de modelo para evitar o teste de um modelo no mesmo conjunto de dados no qual ele foi treinado. Este é um erro comum, especialmente que um conjunto de dados de teste separado nem sempre está disponível. No entanto, isso geralmente leva a medidas de desempenho imprecisas (já que o modelo terá uma pontuação quase perfeita, pois está sendo testado nos mesmos dados em que foi treinado). Para evitar esse tipo de erro, a validação cruzada é geralmente preferida.

O conceito de validação cruzada é realmente simples: em vez de usar todo o conjunto de dados para treinar e, em seguida, testar nos mesmos dados, poderíamos dividir aleatoriamente nossos dados em conjuntos de dados de treinamento e teste.

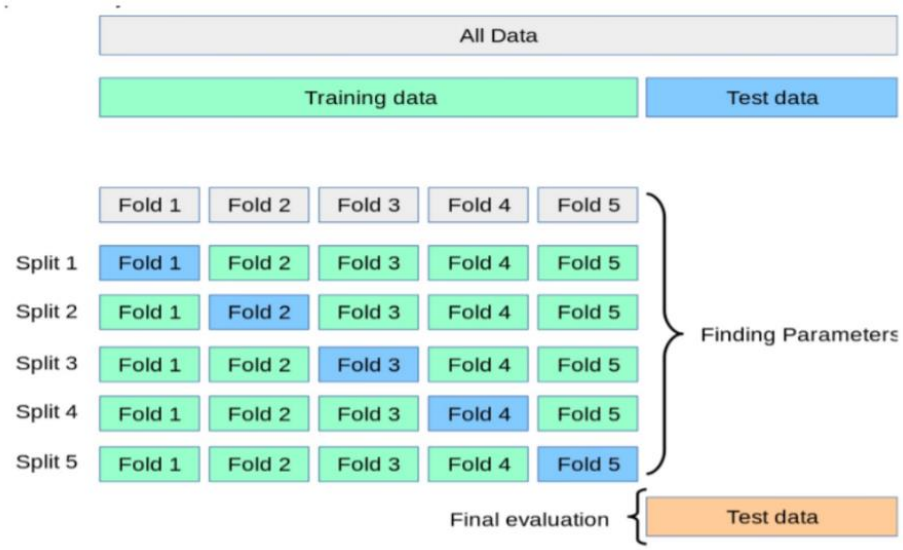
Existem vários tipos de métodos de validação cruzada (LOOCV – validação cruzada de saída única, o método *holdout*, validação cruzada k-fold). Aqui, é discutido o método de validação cruzada *K-fold.*

O *k-fold* consiste, basicamente, nas seguintes etapas (Borra, S; Ciaccio, A, 2010):

1. Divida aleatoriamente os dados em k subconjuntos, também chamados de dobras.
2. Encaixe o modelo nos dados de treinamento (ou k-1 dobras).
3. Use a parte restante dos dados como conjunto de testes para validar o modelo. (Normalmente, nesta etapa, a precisão ou o erro de teste do modelo é medido).
4. Repita o procedimento k vezes.

Dado um conjunto de dados, normalmente a distribuição entre treino e teste e feita conforme a Figura 9, da seguinte forma:

Figura 9 - Validação Cruzada para 5 pastas



Fonte: https://www.rockyourcode.com/ (2021)

3 MÉTODO E MATERIAIS UTILIZADOS

Neste capítulo serão descritos os procedimentos metodológicos e os materiais necessários para solução do problema proposto. Serão apresentadas as ferramentas de software, estratégia de partição do conjunto de treinamento do modelo de classificação e por fim a métrica de avaliação utilizada.

3.1 FERRAMENTAS COMPUTACIONAIS

Foram utilizadas duas linguagens de programação neste trabalho, a linguagem *Python* através da plataforma de programação *Google Colaboratory*, e a plataforma *MATLAB,* a fim de fazer a comparação com os resultados encontrados através do script em *Python*.

3.1.1 GOOGLE COLABORATORY

A Google criou o *Google Colaboratory* ou simplesmente *Colab* para facilitar e difundir o ensino e a pesquisa em aprendizado de máquina. O *Colab* é um ambiente virtual na nuvem da Google onde o programador tem acesso gratuito a um *Jupyter notebook*. O projeto tem o nome Jupyter pois foi originalmente criado para o ensino das linguagens JUlia, PYThon e R. Posteriormente, o projeto foi estendido e suporta várias linguagens. Este ambiente é executado em um navegador e oferece vários recursos interessantes para ensino e pesquisa.

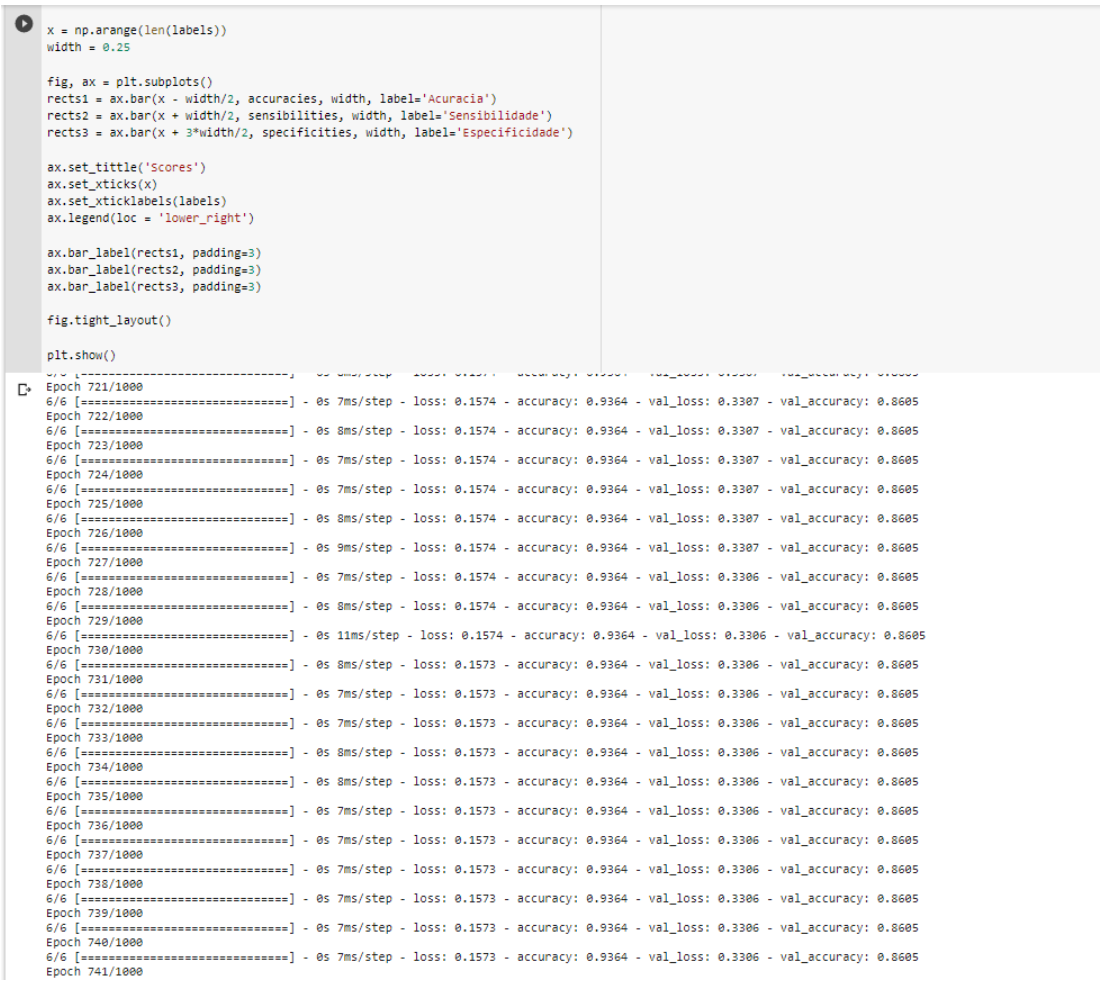
Ao criar uma sessão no *Colab*, o usuário tem acesso a um processador com dois núcleos, 12 GBytes de memória RAM e cache L3 de 40-50 Mbytes. Além disso, o usuário pode ter acesso a uma TPU ou uma GPU. No caso de abrir uma nova sessão, o usuário deve configurar o ambiente de execução se for utilizar a TPU ou a GPU como acelerador. Este recurso é importante para executar os códigos de aprendizado de máquina. O usuário também tem acesso a um sistema de arquivos com uma capacidade de armazenamento que varia de 30 a 300 Gbytes de espaço em disco e um terminal *Linux*.

O Google Colaboratory é amplamente utilizado para executar código em Python com as bibliotecas e as ferramentas de aprendizado de máquina. O ambiente tem dois

tipos de células, as de texto e as células de código. Nativamente, a célula de código interpreta e executa Python. Os resultados podem ser visualizados logo abaixo da célula ao usar o comando print.

A Figura 10 mostra a compilação do script feito com o treinamento do modelo, executado abaixo da célula do código.

Figura 10 - Resultado de compilação.



```
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, accuracies, width, label='Acuracia')
rects2 = ax.bar(x + width/2, sensibilities, width, label='Sensibilidade')
rects3 = ax.bar(x + 3*width/2, specificities, width, label='Especificidade')

ax.set_tittle('Scores')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend(loc = 'lower_right')

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)
ax.bar_label(rects3, padding=3)

fig.tight_layout()

plt.show()
```

```
Epoch 721/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3307 - val_accuracy: 0.8605
Epoch 722/1000
6/6 [==============================] - 0s 8ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3307 - val_accuracy: 0.8605
Epoch 723/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3307 - val_accuracy: 0.8605
Epoch 724/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3307 - val_accuracy: 0.8605
Epoch 725/1000
6/6 [==============================] - 0s 8ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3307 - val_accuracy: 0.8605
Epoch 726/1000
6/6 [==============================] - 0s 9ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3307 - val_accuracy: 0.8605
Epoch 727/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 728/1000
6/6 [==============================] - 0s 8ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 729/1000
6/6 [==============================] - 0s 11ms/step - loss: 0.1574 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 730/1000
6/6 [==============================] - 0s 8ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 731/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 732/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 733/1000
6/6 [==============================] - 0s 8ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 734/1000
6/6 [==============================] - 0s 8ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 735/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 736/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 737/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 738/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 739/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 740/1000
6/6 [==============================] - 0s 7ms/step - loss: 0.1573 - accuracy: 0.9364 - val_loss: 0.3306 - val_accuracy: 0.8605
Epoch 741/1000
```

Fonte: Autoria própria (2021)

3.1.1.1 BIBLIOTECAS UTILIZADAS

- Biblioteca *Numpy*: A biblioteca *NumPy* é fundamental para a computação científica em Python. Através desta biblioteca e possível manipular objetos do

tipo array, com múltiplas dimensões, incluindo suas variações como é o caso das matrizes. Além disso, são fornecidas operações de álgebra linear, estatística e muitas outras.

- Biblioteca Pandas: Pandas é uma biblioteca de código livre que fornece métodos práticos de se operar com estruturas de dados. Para o estudo de ciência de dados, é especialmente útil na manipulação de datasets. Dentre os recursos oferecidos estão: redimensionamento, agrupamento, substituições, entre outros. As extensões de arquivos permitidas incluem os formatos ".xlsx", ".xls", ".csv', por exemplo.
- Biblioteca *Matplotlib*: *Matplotlib* é uma biblioteca para visualização de dados estatísticos e criação de gráficos. *Pyplot* e um módulo do matplotlib que fornece uma interface semelhante ao *Matlab*, com a vantagem de ser de código aberto e totalmente gratuito. Esta biblioteca pode ser integrada a scripts desenvolvidos na linguagem Python.
- Biblioteca *Tensorflow*: *TensorFlow* é uma biblioteca de código aberto para aprendizado de máquina aplicável a uma ampla variedade de tarefas. É um sistema para criação e treinamento de redes neurais para detectar e decifrar padrões e correlações.

3.1.2 MATLAB

O *MATLAB* é um software de alto desempenho destinado a fazer cálculos com matrizes (*MATrix LABoratory*), podendo funcionar como uma calculadora ou como uma linguagem de programação científica. Entretanto, os comandos do *MATLAB* são mais próximos da forma como escrevemos expressões algébricas, tornando mais simples o seu uso. Atualmente, o *MATLAB* é definido como um sistema interativo e uma linguagem de programação para computação técnica e científica em geral, integrando a capacidade de fazer cálculos, visualização gráfica e programação (Tonini e Couto, 1999).

- Uso típico do *MATLAB*:
  - Cálculos matemáticos;
  - Desenvolvimento de algoritmos;
  - Modelagem, simulação e confecção de protótipos;

- Análise, simulação e confecção de dados;
- Gráficos científicos e de engenharia;
- Desenvolvimento de aplicações, incluindo a elaboração de interfaces gráficas com o usuário.

No gerenciador de programas do *Windows*, um duplo clique no ícone *MATLAB* carrega o aplicativo. Uma vez inicializado o *MATLAB*, aparecerá na tela uma janela de comandos e o *prompt* padrão (EDU>> ou >>) é exibido na tela. A partir deste ponto, o software espera um comando (instruções) do usuário. Todo comando deve ser finalizado teclando-se Enter.

O MATLAB tem uma série de funções científicas pré-definidas. A palavra função tem um significado diferente daquele que tem na Matemática. A função é um comando, que pode ter alguns argumentos de entrada e alguns de saída. Algumas dessas funções são intrínsecas, ou seja, não podem ser alteradas pelo usuário. Outras funções estão disponíveis em uma biblioteca externa distribuídas com o programa original (*MATLAB TOOLBOX*), que são na realidade arquivos com a extensão ".m" criados a partir das funções intrínsecas. A biblioteca externa pode ser constantemente atualizada à medida que novas aplicações são desenvolvidas. As funções do *MATLAB*, intrínsecas ou arquivos ".m", podem ser utilizadas apenas no ambiente.

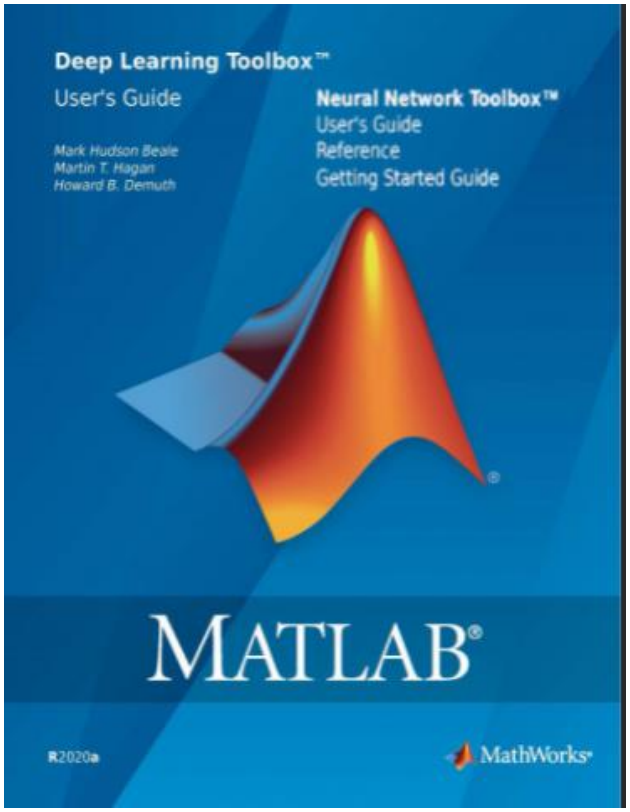
As categorias gerais de funções matemáticas disponíveis no *MATLAB* incluem:

- Matemática elementar;
- Funções especiais;
- Matrizes elementares e especiais;
- Decomposição e faturização de matrizes;
- Análise de dados;
- Polinômios;
- Solução de equações diferenciais;
- Equações não-lineares e otimização;
- Integração numérica;
- Processamento de sinais.

Levando em conta as informações citadas acima, foi feito um script simplificado no *MATLAB* utilizando as fórmulas matemáticas do modelo proposto, por se tratar de uma plataforma computacional matemática, a fim de mostrar se existe

diferença entre seus resultados e os encontrados utilizando o *Google Colab*. Pata este trabalho, foi utilizado o *MATLAB* 2020, como mostra a imagem abaixo.

Figura 11 - MATLAB 2020



Fonte: Autoria Própria (2021)

3.2 DESCRIÇÃO DO PROBLEMA PROPOSTO E BASE DE DADOS

Este trabalho tem a finalidade de implementar o modelo de regressão logística contendo uma camada, aplicado a classificação de exames de espectrometria de massa na previsão de câncer de ovário. Para isso foramutilizadas as bases de dados "*ovarianInputs*" e "*ovarianTargets*". Essas bases de dados foram originalmente obtidas no programa de Proteômica Clínica FDA-NCI publicado pelo Centro para Pesquisa do Câncer do Instituto Nacional do Câncer. O MATLAB utiliza uma amostra dessa base de dados em seu site oficial, aplicando em exemplos para algoritmos de aprendizado de máquina.

Os dados são provenientes de exames de espectrometria de massa e contêm informações relativas a 100 intensidades de íons medidas para 100 diferentes valores

específicos de massa-carga, para 216 pacientes diferentes. A base de dados é constituída por dois arquivos, descritos a seguir:

- *ovarianInputs* - uma matriz 216x100 que define 100 níveis de intensidade de íons medido para 100 diferentes valores específicos de massa-carga, para 216 pacientes diferentes.
- *ovarianTargets* - uma matriz 216x2 onde cada elemento indica um paciente com câncer de ovário com [1; 0] ou paciente normal com [0; 1]. Existem 121 pacientes com câncer e 95 pacientes normais.

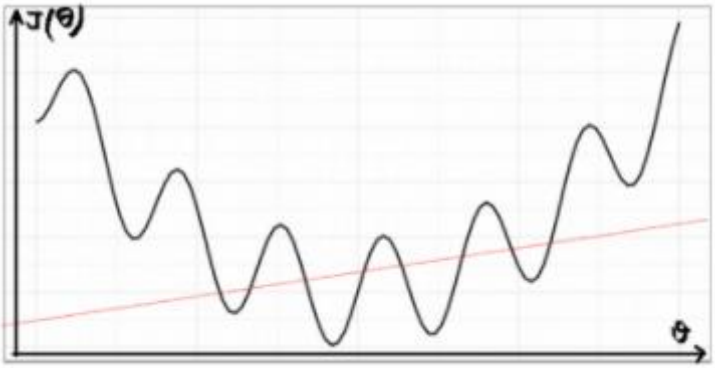
Neste trabalho, os 100 atributos de cada observação serão utilizados como a primeira camada da Rede Neural Artificial, modelada pelo Regressor Logístico. Com este modelo objetiva-se diagnosticar determinado paciente como portador de câncer, baseando-se nos dados obtidos no seu exame de espectrometria. Este modelo deve ser treinado e testado de acordo com o método de validação cruzada, ou seja, o dataset será dividido em 5 pastas distintas, estando estes conjuntos particionados em subconjunto de treino e subconjunto de teste. Como métrica de avaliação, para cada conjunto de teste, será elaborada uma matriz de confusão, visando extrair a acurácia local. Ao final do experimento, devem ser apresentados os valores de acurácia, especificidade e sensibilidade.

3.3 AJUSTE MATEMÁTICO DO MODELO

No método de Regressão linear, geralmente, utiliza-se a função quadrática, dada a sua convexidade e, portanto, a convergência do algoritmo para um ponto mínimo único.

Entretanto, para o problema da classificação binária, proposto nesse trabalho, caso essa função fosse aplicada a um algoritmo de regressão logística, a não linearidade produzida pela função sigmoide no cálculo das predições $\pi(x)$ comprometeria a convergência do algoritmo devido a introdução de vários mínimos locais, como mostra a Figura 12.

Figura 12 - Função não convexa



Fonte: https://ichi.pro/pt/regressao-logistica-131466354476160 (2021)

Para evitar esse problema, foi formulada na equação 13 a hipótese $h_\theta(x) = \pi(x)$ de um conjunto de parâmetros θ associados a um conjunto de variáveis preditoras x_i, de tal forma que a saída esteja no intervalo de $[0,1]$, em que $g(\theta)$ representa a função sigmoide:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{(-\theta^T x)}} \tag{13}$$

Tal hipótese representa a probabilidade estimada de que a saída y_i seja igual a 1 quando submetida à entrada x_i parametrizada por θ_i:

$$h_\theta(x) = P(y = 1|x;\theta) = 1 - P(y = 0|x;\theta)$$

Para tornar a função de custo convexa (garantir a convergência para o mínimo global), aplicou-se uma transformada usando o logaritmo da função sigmoide, conforme a equação 14:

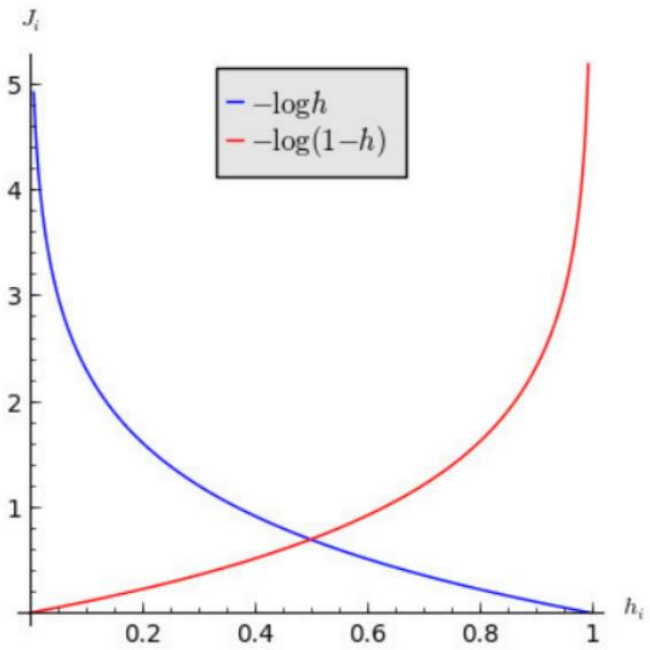
$$Erro(h_\theta(x), y) = \begin{Bmatrix} -log(h_\theta(x)), & p/\ y = 1 \\ -\log\big(1 - (h_\theta(x)\big), & p/\ y = 0 \end{Bmatrix} \tag{14}$$

Essa função de custo pode então ser representada por uma única equação, dada por:

$$Erro(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)(\log\,(1 - h_\theta(x))) \tag{15}$$

A Figura 13 mostra o gráfico dessa função destacando sua convexidade para as 2 classes binárias y_i $[0, 1]$.

Figura 13 - Função de custo logística



Fonte: https://ichi.pro/pt/regressao-logistica-131466354476160 (2021)

Definida a função de custo, podemos adotar o método do Gradiente Descendente, abordado no capítulo 2, para otimizar e encontrar os valores dos coeficientes θ e w_i.

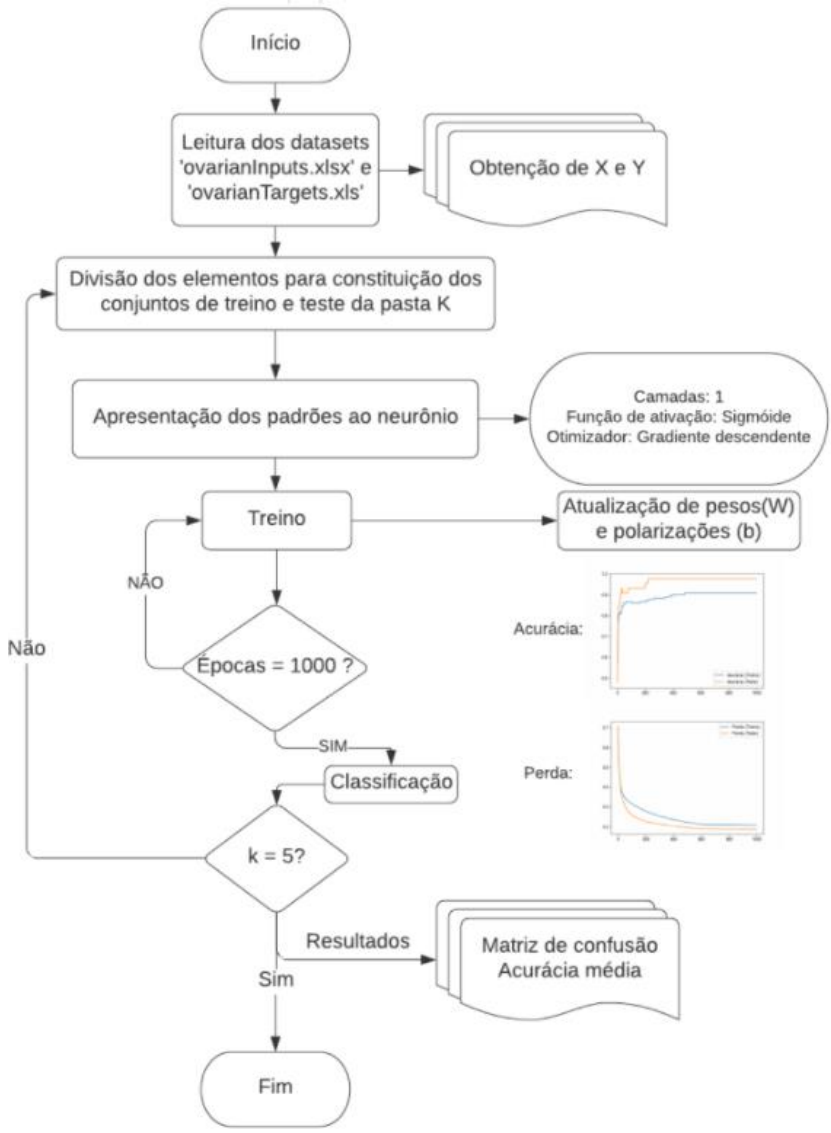
3.4 IMPLEMENTAÇÃO DO MODELO

A implementação foi realizada através do desenvolvimento de um script na linguagem *Python*, tendo o suporte dos pacotes *Numpy*, *Pandas* e *Tensorflow*. Para a edição e execução do código foi utilizado o ambiente de desenvolvimento *Google Colaboratory*. Para fins de comparação dos resultados, o mesmo modelo foi implementado no software Matlab com o suporte das definições matemáticas apresentadas nos capítulos 2 e 3.

O fluxograma exibido na Figura 14 apresenta o procedimento completo de treinamento da máquina, obtendo a cada pasta a acurácia de classificação para o conjunto de

dados apresentados. Por fim, obteve-se a média das acurácias alcançadas em cada uma das pastas, bem como a sua respectiva matriz de confusão, a fim de analisar o desempenho médio do classificador.

Figura 14 - Fluxograma do algoritmo



Fonte: Autoria Própria (2021)

A seguir serão comentadas detalhadamente as principais partes de implementação e programação.

Inicialmente foi realizada a leitura do dataset através do método *read_excel* do pacote *Pandas*, seguida da inserção de uma coluna adicional com o valor 1 para fins de polarização, como mostra o trecho de código da Figura 15.

Figura 15 - Leitura dos dados e polarização

```
#Leitura do banco de dados e variáveis alvo
data = pd.read_excel('ovarianInputs.xlsx', header = None)
target_data = pd.read_excel('ovarianTargets.xls', header = None)

#Definir o label do gráfico da matriz de confusão
graph_labels = ['Cancer', 'Não - Câncer']

#Data
x = np.array(data).reshape((-1, 100))
ones_columm = np.ones(216, dtype=float). reshape(-1, 1)
x = np.concatenate([x, ones_columm], axis = 1)
```

Fonte: Autoria Própria (2021)

O treinamento ocorreu em 5 pastas, sendo utilizado o método de validação cruzada, conforme Figura 16, com o intuito de variar o conjunto de treinamento ao longo de todo o dataset, e avaliar qual possui melhor desempenho.

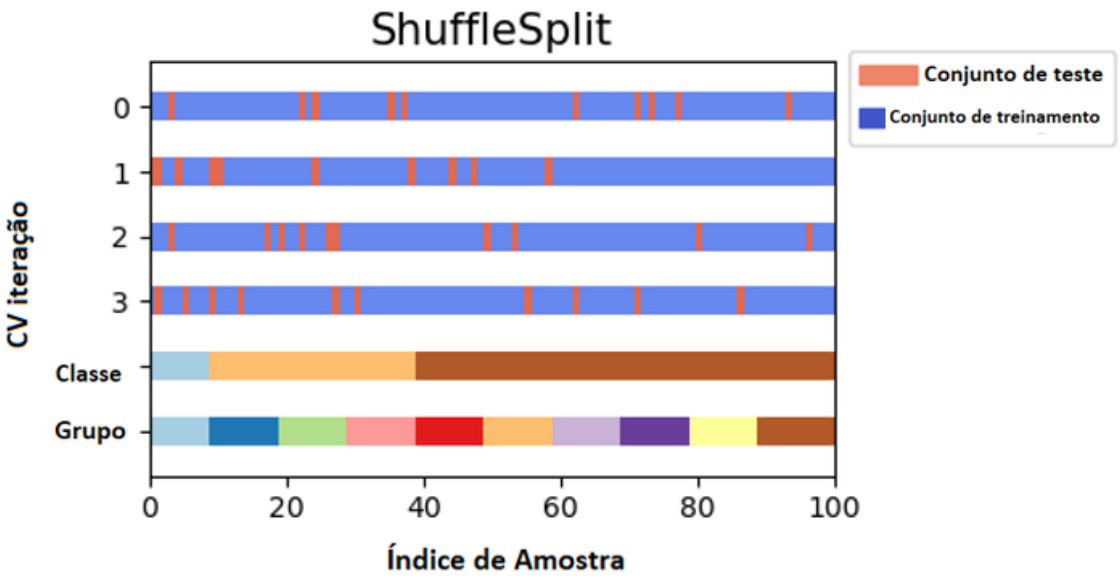
Figura 16 - Treinamento por validação cruzada

```
#Define cross-validation K-fold
kfold = KFold(n_splits=5, shuffle=True)
#skf = StratifiedKFold(n_splits=5)

#K-fold Cross Validation model evaluation
fold_no = 1

accuracies = list()
sensibilities = list()
specificities = list()

for train, test in kfold.split(x, y):
  X_train, X_test, y_train, y_test = x[train], x[test], y[train], y[test]

  N, D = X_train.shape
  #escalonamento [0,1]

  scaler = StandardScaler()
  X_train = scaler.fit_transform(X_train)
  X_test = scaler.transform(X_test)

  print(X_train)
  print(X_test)
  #input('break)
```

Fonte: Autoria própria (2021)

Estando o conjunto de entrada com quantidades desiguais de classes, sendo a classe 1 majoritária, com 121 observações, adotou-se o procedimento de *shuffle*, isto é, dentro de uma pasta os dados foram coletados aleatoriamente de forma a não ocorrer predominância de uma classe ou outra. A Figura 17 ilustra essa metodologia de segmentação de dados.

Figura 17 - Método Shuffle Split – Divisão aleatória



Fonte: adaptado de https://scikit-learn.org/ (2021)

Ao utilizar 5 pastas, o conjunto de treinamento constituiu-se de 80% dos dados, o que corresponde a 173 observações, restando 46 observações para o conjunto de teste.

Para a realização do fluxo de aprendizado de máquina, foram utilizadas algumas etapas fundamentais, a citar:

- Definição de cada Neurônio: Na ferramenta Tensorflow esta funcionalidade foi implementada pela função ***tf.keras.Sequential***, conforme Figura 18. Para esta aplicação definiu-se o neurônio com 1 camada, sendo a função de ativação a sigmoide.

Figura 18 - Definição de cada neurônio

```
#Define o modelo da primeira camada de neuronio
model = tf.keras.models.Sequential ([
  tf.keras.layers.Input(shape=(D,)),
  tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Fonte: Autoria própria (2021)

- Definição da função de otimização: Na ferramenta Tensorflow esta funcionalidade foi implementada através do método ***model.compile***, com o parâmetro 'SGD', indicando que se trata do gradiente descendente, como mostra a Figura 19.

Figura 19 - Função de otimização e função de perda

```
#utilizar a função de perda logística
model.compile(optimizer='SGD',
              loss='binary_crossentropy',
              metrics=['accuracy'])
scheduler = tf.keras.callbacks.LearningRateScheduler(schedule)
```

Fonte: Autoria própria (2021)

- Definição da função de perda: Ainda no método ***model.compile***, utilizou-se o parâmetro ***binary_crossentropy***, equivalente ao logaritmo da função perda.

- Treinamento do modelo: Realizado pela função ***model.fit***, parametrizada com o conjunto de treino e número de épocas, como mostra a Figura 20.

Figura 20 - Treinamento do modelo

```
#Treinando o modelo
r = model.fit(X_train, y_train, validation_data=(X_test, y_test)
, epochs=1000, callbacks = [scheduler])

aux = model.layers[0].get_weights()
#Pesos
print(aux[0])
```

Fonte: Autoria própria (2021)

- Por fim, teste do modelo e obtenção de métricas, conforme Figura 21.

Figura 21 - Trecho de teste do modelo e métricas de avaliação

```
  #Evaluate the model - evaluate() returns loss and accuracy
  print("Train_score: ", model.evaluate(X_train, y_train))
  print("Test_score: ", model.evaluate(X_test, y_test))

  print('Making_predictions')
  count = 0
  sensibility = 0
  specificity = 0
  P = model.predict(X_test)
  P = np.round(P).flatten()
  for j, element in enumerate(P):
    if element == y_test[j]:
      count = count+1
      if element == 0:
        specificity = specificity + 1
      else:
        sensibility = sensibility + 1

true_quantity = np.count_nonzero(y_test == 1)
false_quantity = np.count_nonzero(y_test == 0)

acc = round(count/len(P), 4)
sens = round(sensibility/true_quantity, 4)
spec = round(specificity/false_quantity, 4)

#Calculate de accuracy, compare it to evaluate() output
print("Manually_calculated_accuracy:", acc)
print("Manually_calculated_sensibility:", sens)
print("Manually_calculated_specificity:", spec)
print("Pacientes_com_cancer: _(y_true)_", true_quantity)
print("Pacientes_sem_cancer: _(y_true)_", false_quantity)

testing = np.concatenate([y_test, np.array(P).reshape((-1, 1))], axis = 1)
print(testing)

accuracies.append(acc)
sensibilities.append(sens)
specificities.append(spec)
```

Fonte: Autoria própria (2021)

O treinamento do modelo, tanto na ferramenta TensorFlow quanto no Matlab, foi realizado com uma taxa de aprendizado adaptativa, isto é, a partir de 600 épocas diminuiu-se por um fator de 10.

4 RESULTADOS E DISCUSSÕES

A seguir serão mostrados os resultados propostos no trabalho.

Dado que o processo de validação cruzada foi realizado de maneira randômica, existem diversas performances possíveis de serem obtidas com o modelo utilizado. Dentre os experimentos realizados, ao todo foram 5, a tabela 1 exibe aqueles cujas pastas obtiveram melhor êxito para a solução baseada na biblioteca *Tensorflow*, considerando as métricas propostas.

Tabela 1 - Desempenho com Tensorflow

Métrica	Experi. 1	Experi. 2	Experi. 3	Experi. 4	Experi. 5
Acurácia	97,73%	93,02%	93,02%	97,67%	83,72%
Sensibilidade	95,83%	96,15%	82,35%	96%	75,86%
Especificidade	100%	88,24%	100%	100%	100%
Acur. Média	93,03%				
Sens. Média	89,24%				
Espec. Média	97,65%				

Fonte: Autoria própria (2021)

A tabela 1 mostra que a acurácia máxima foi encontrada no experimento 1 - pasta 1, entretanto, todas os experimentos obtiveram resultados satisfatoriamente próximos, levando-os a ter uma acurácia média de 93,03%; sensibilidade média de 89,24% e especificidade média de 97,65%.

A tabela 2 mostra os resultados encontrados utilizando a ferramenta MATLAB.

Tabela 2 - Desempenho com Matlab

Métrica	Experi. 1	Experi. 2	Experi. 3	Experi. 4	Experi. 5
Acurácia	95,34%	90,69%	93,02%	95,34%	93,02%
Sensibilidade	92,00%	84%	88%	92%	88%
Especificidade	100%	94,74%	94,74%	100%	94,74%
Acur. Média	93,48%				
Sens. Média	88,80%				
Espec. Média	96,84%				

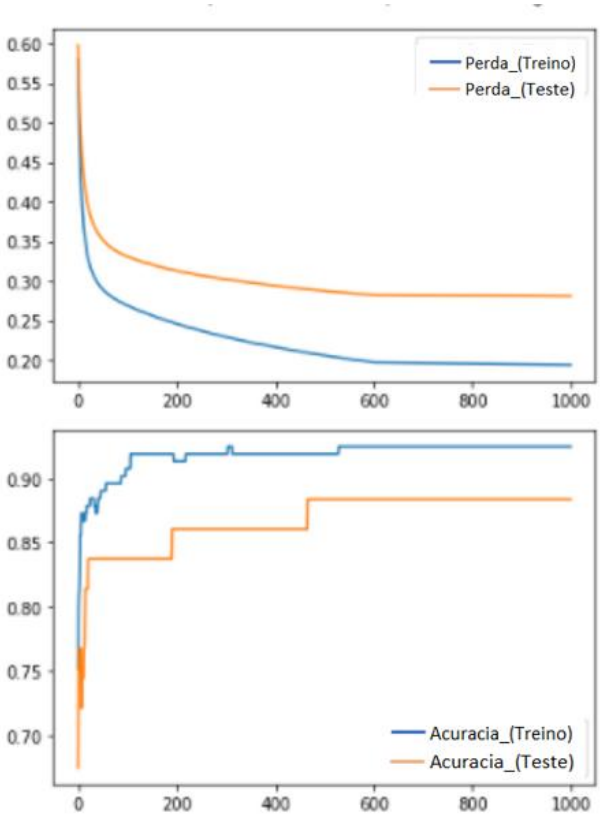
Fonte: Autoria própria (2021)

Conforme a Tabela 2, utilizando o Matlab, a acurácia máxima foi obtida nos experimentos 1 e 4 – Pasta 1. Todos os experimentos tiveram uma acurácia média de 93,48%, sensibilidade média de 88,80% e 96,84%.

Os resultados obtidos tanto no Matlab como usando *tensorflow* foram parecidos, mostrando que a plataforma matemática do Matlab não leva vantagem sobre as bibliotecas do *Phyton* para *Machine Learning*, mostrando ainda que o *Google Colab* é a principal ferramenta da atualidade para pesquisa e programação na área de aprendizado de máquina e Redes Neurais pois apresenta maior facilidade e rapidez na criação de experimentos. O Matlab, por sua vez, é uma ferramenta matemática poderosíssima, útil para experimentos que necessitam de uma maior usabilidade matemática.

A Figura 22 mostra respectivamente as curvas de treino e teste para perda e acurácia do experimento 1.

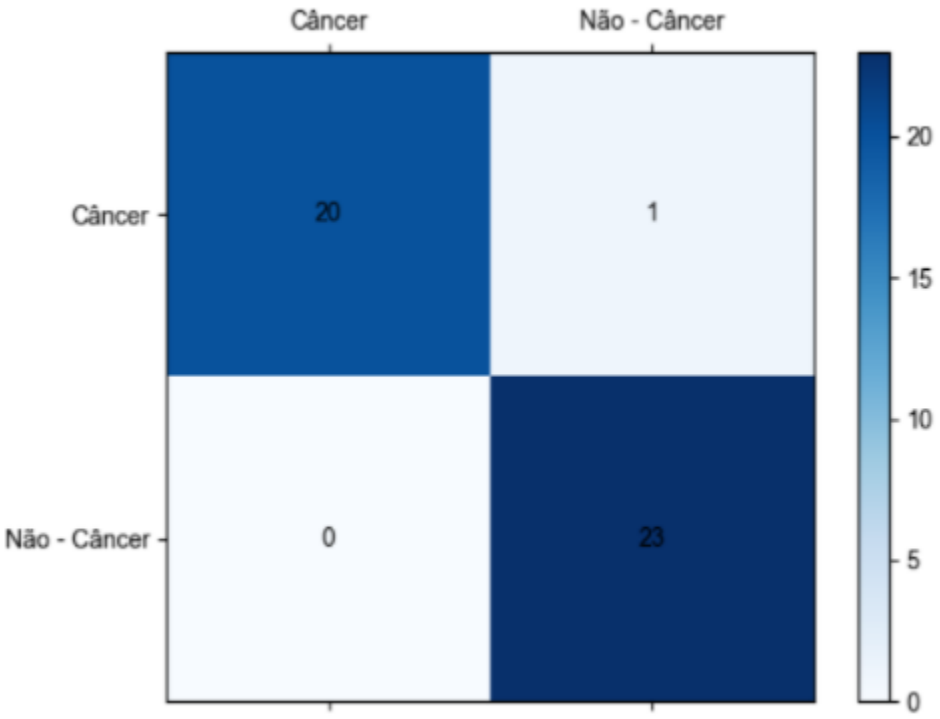
Figura 22 - Curvas de perda e acurácia para treinamento e teste



Fonte: Autoria própria (2021)

Como maneira de expressar graficamente o resultado das classificações e facilitar o processo de análise dos resultados, foram geradas as matrizes de confusão para cada pasta através dos métodos da biblioteca matplotlib. A Figura 23 ilustra a matriz de confusão da do experimento 1 utilizando *tensorflow*, cujo valor de acurácia superou as demais.

Figura 23 - Matriz de confusão experimento 1 – Pasta 1

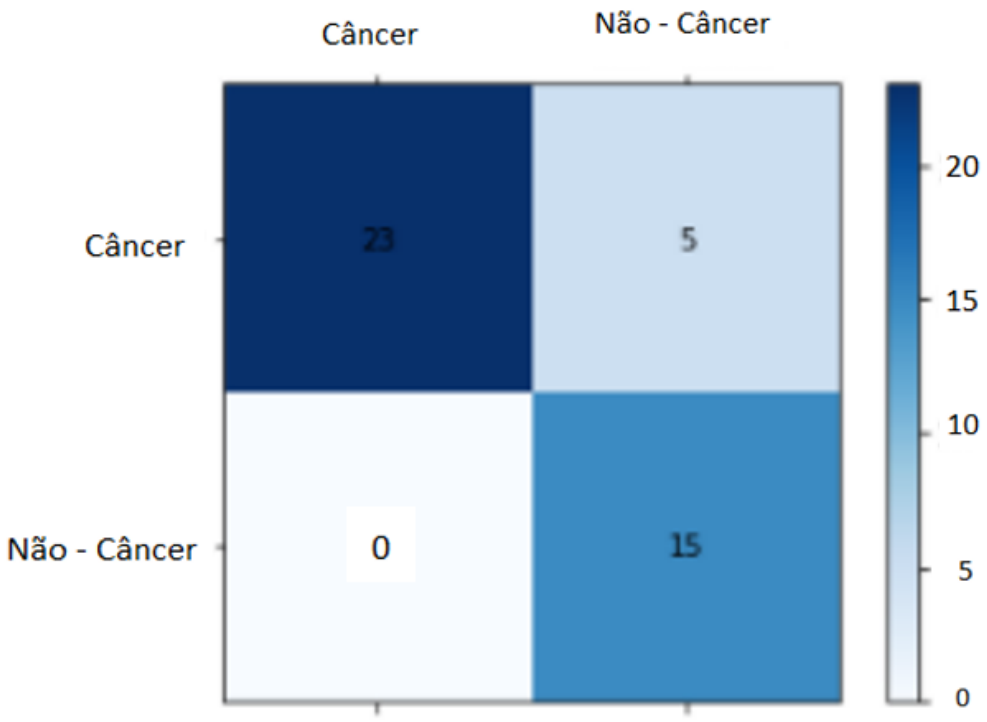


Fonte: Autoria própria (2021)

A matriz de confusão com acurácia de 97,73% pode ser interpretada da seguinte maneira: 20 pacientes com certeza têm câncer, 1 paciente pode ou não ter câncer e 23 pacientes com certeza não tem câncer, teoricamente a matriz prova a grande eficácia do experimento.

A fim de fazer a comparação da matriz de confusão com outro resultado aleatório, a Figura 24 traz a matriz de confusão do quinto experimento, o que obteve menor acurácia dentre todos, com acurácia média de 83,372%, e mostra que: 23 pacientes com certeza não têm câncer, 5 pacientes podem ou não ter câncer, e 15 não tem câncer.

Figura 24 - Matriz de confusão experimento 5 teste



Fonte: Autoria própria (2021)

5 CONCLUSÃO

O modelo logístico é frequentemente utilizado em situações em que a variável resposta é de natureza binária (sucesso e fracasso), na qual seus valores são expressos em termos de probabilidade. Para entender melhor o contexto de regressão logística, foi feita uma abordagem sobre redes neurais e aprendizado de máquina no Capítulo 2, para que em seguida, a teoria do modelo logístico fosse mostrada.

As métricas são essenciais para uma avaliação do modelo ajustado, pois a partir deles pode-se ter conhecimento da sua capacidade de ajuste, sensibilidade, especificidade e matriz de confusão. Tais resultados normalmente são utilizados como critério para o ponto de corte na classificação da variável de interesse. Logo, a partir de um determinado valor de probabilidade, a variável resposta é classificada como sucesso.

O principal objetivo desta aplicação foi ajustar um modelo logístico capaz de prever se pacientes tem ou não câncer de ovário a partir de resultados de exames de espectrometria previamente feitos. O modelo foi adquirido a partir de um estudo com base de dados constituída por 216 observações, das quais representam resultados de pacientes. Apesar da amostra de dados ser relativamente pequena para um estudo de regressão, seus resultados foram altamente satisfatórios.

A implementação do regressor logístico para a rede neural de 1 camada, teve como característica a simplicidade de implementação na ferramenta *Tensorflow*, possibilitando agilidade para a validação de modelos de redes neurais com diversas topologias e camadas. Da mesma maneira, com os conceitos teóricos vistos foi possível desenvolver um algoritmo baseado na rede de propagação direta com o apoio do software *Matlab*, uma vez que as funções de custo, de ativação e gradiente descente puderam ser modeladas matematicamente com os recursos da plataforma.

O método de validação cruzada seguiu os mesmos princípios abordados em trabalhos anteriores, adicionando o fato de que os conjuntos de teste e treinamento foram tomados de maneira aleatória, dado que a quantidade de observações para cada classe não é a mesma. Os cinco experimentos realizados obtiveram as métricas exibidas na Tabela 1.

As acurácias médias de 93,03% e máxima de 97,73% indicam um nível de assertividade aceitável, visto que para a pasta 1 exibida na Figura 23, apenas 1 de 44 itens foi classificado erroneamente. Foi possível observar que o modelo gerado possuiu alto índice de especificidade para todas as pastas, indicando que apresenta confiabilidade em classificar pacientes que não possuem câncer.

Portanto, o regressor logístico mostra-se robusto e de boa performance para dados reais, com a vantagem de possuir baixo custo computacional. O modelo apresenta perspectiva de melhorias para um conjunto maior de dados e aumento no número de camadas.

6 TRABALHOS FUTUROS

Para o aperfeiçoamento deste trabalho, outras técnicas de *Machine Learning* podem ser aplicadas, a fim de obter-se resultados de acurácia maiores em sua totalidade. Também podem ser aplicadas uma maior base de dados para treinamento e teste do modelo com o intuito de aumentar o range de assertividade.

Também podem ser aplicadas diferentes métricas de avaliação adicionais, como a curva ROC – *Receiver Operating Characteristic*, que ilustra a relação entre sensibilidade e especificidade, e pode ser utilizada para decidir um bom ponto de corte.

Ademais, as técnicas de *Machine Learning* em geral podem ser utilizadas a fim de ajudar pesquisas voltadas a área médica.

APÊNDICE

ANEXO 1 – SCRIPT PHYTON

```
import tensorflow as tf
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler , MinMaxScaler
from sklearn.model_selection import KFold , StratifiedKFold
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

#learning rate scheduler
def schedule (epoch, lr, logs = {}):
  if epoch >= 600:
    return 0.0001
  return 0.001

#Leitura do banco de dados e variáveis alvo
data = pd.read_excel('ovarianInputs.xlsx', header = None)
target_data = pd.read_excel('ovarianTargets.xls', header = None)

#Definir o label do gráfico da matriz de confusão
graph_labels = ['Cancer', 'Não - Câncer']

#Data
x = np.array(data).reshape((-1, 100))
ones_columm = np.ones(216, dtype=float). reshape(-1, 1)
x = np.concatenate([x, ones_columm], axis = 1)

#Targets
y = np.array(target_data[0]). reshape((-1, 1))

#Define cross-validation K-fold
kfold = KFold(n_splits=5, shuffle=True)
#skf = StratifiedKFold(n_splits=5)

#K-fold Cross Validation model evaluation
fold_no = 1

accuracies = list()
sensibilities = list()
specificities = list()

for train, test in kfold.split(x, y):
```

```
  X_train, X_test, y_train, y_test = x[train], x[test], y[train], y[tes
t]

  N, D = X_train.shape
  #escalonamento [0,1]

  scaler = StandardScaler()
  X_train = scaler.fit_transform(X_train)
  X_test = scaler.transform(X_test)

  print(X_train)
  print(X_test)
  #input('break)

  #Define o modelo da primeira camada de neuronio
  model = tf.keras.models.Sequential ([
    tf.keras.layers.Input(shape=(D,)),
    tf.keras.layers.Dense(1, activation='sigmoid')
  ])

  #utilizar a função de perda logística
  model.compile(optimizer='SGD',
                loss='binary_crossentropy',
                metrics=['accuracy'])
  scheduler = tf.keras.callbacks.LearningRateScheduler(schedule)

  #Treinando o modelo
  r = model.fit(X_train, y_train, validation_data=(X_test, y_test)
  , epochs=1000, callbacks = [scheduler])

  aux = model.layers[0].get_weights()
  #Pesos
  print(aux[0])

  #Evaluate the model - evaluate() returns loss and accuracy
  print("Train_score: ", model.evaluate(X_train, y_train))
  print("Test_score: ", model.evaluate(X_test, y_test))

  print('Making_predictions')
  count = 0
  sensibility = 0
  specificity = 0
  P = model.predict(X_test)
  P = np.round(P).flatten()
  for j, element in enumerate(P):
    if element == y_test[j]:
      count = count+1
      if element == 0:
        specificity = specificity + 1
```

```
      else:
        sensibility = sensibility + 1

true_quantity = np.count_nonzero(y_test == 1)
false_quantity = np.count_nonzero(y_test == 0)

acc = round(count/len(P), 4)
sens = round(sensibility/true_quantity, 4)
spec = round(specificity/false_quantity, 4)

#Calculate de accuracy, compare it to evaluate() output
print("Manually_calculated_accuracy:", acc)
print("Manually_calculated_sensibility:", sens)
print("Manually_calculated_specificity:", spec)
print("Pacientes_com_cancer: _(y_true)_", true_quantity)
print("Pacientes_sem_cancer: _(y_true)_", false_quantity)

testing = np.concatenate([y_test, np.array(P).reshape((-
1, 1))], axis = 1)
print(testing)

accuracies.append(acc)
sensibilities.append(sens)
specificities.append(spec)

#Bloco para estilização e vizualização da matriz de confusão
window_name = 'Pasta_' + str(fold_no) + '_Matriz_de_confusão'
fig = plt.figure(window_name)
matrix = confusion_matrix(P, y_test)
print(matrix)
ax = fig.add_subplot(111)
cax = ax.matshow(matrix, cmap = 'Blues')
for(m, n), label in np.ndenumerate(matrix):
  ax.text(n, m, int(label), ha='center', va='center')
ax.set_xticklabels(['']+graph_labels)
ax.set_yticklabels(['']+graph_labels)
fig.colorbar(cax)

#Plot Whats returned by model.fit()
window_name = 'Pasta_' + str(fold_no) + '_Perdas'
fig = plt.figure(window_name)
plt.plot(r.history['loss'], label='Perda_(Treino)')
plt.plot(r.history['val_loss'], label='Perda_(Teste)')
plt.legend()

#Plot the accuracy
window_name = 'Pasta_' + str(fold_no) + '_Acuracias'
fig = plt.figure(window_name)
plt.plot(r.history['accuracy'], label='Acuracia_(Treino)')
```

```
plt.plot(r.history['val_accuracy'], label='Acuracia_(Teste)')
plt.legend()

plt.show()

fold_no = fold_no + 1


print('Acuracias:_', accuracies)
print('Sensibilidades:_', sensibilities)
print('ESpecificidades:_', specificities)
print('Media:_', np.mean(accuracies))
print('Maxima:_', np.max(accuracies))

accuracies.append(np.mean(accuracies))
sensibilities.append(np.mean(sensibilities))
specificities.append(np.mean(specificities))

metricas = [accuracies, sensibilities, specificities]
print(metricas)

labels = ['Pasta_1', 'Pasta_2', 'Pasta_3', 'Pasta_4', 'Pasta_5', 'Mean'
]

sns.set_theme(style='whitegrid')

x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, accuracies, width, label='Acuracia')
rects2 = ax.bar(x + width/2, sensibilities, width, label='Sensibilidade
')
rects3 = ax.bar(x + 3*width/2, specificities, width, label='Especificid
ade')

ax.set_tittle('Scores')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend(loc = 'lower_right')

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)
ax.bar_label(rects3, padding=3)

fig.tight_layout()

plt.show()
```

ANEXO 2 – SCRIPT MATLAB

```
close all;
clear all;

X = xlsread('ovarianInputs.xlsx');
R = xlsread('ovarianTargets.xlsx');
Y = R(:, 1);
m = lenght(Y);
[r, c] = size(X);
idx = randperm(m);
%y = Y + 1;
y = Y;
p = 0.8;
wMax = zeros(101);

xtrain = X(idx(1:round(p*m)),:);
ytrain = X(idx(1:round(p*m)),:);
xtest = X(idx(1:round(p*m)+1:lenght(idx)),:);
ytest = X(idx(1:round(p*m)+1:lenght(idx)),:);
Num_Pastas=5;
Acuracia_Media = 0;

for Pasta = 1:Num_Pastas

%atualiza w pelo gradiente descendente
alpha=0.001;
w=double(randperm(c+1)/100);
w=-w';__
mtrain=lenght(ytrain);
xtrain2[ones(mtrain, 1) xtrain];
Acuracia_TrainingSet = 0.0;

cont = 1;
```

```
epoch = 1;
while (epoch <= 1200)
    if epoch >= 600
        alpha = 0.0001;
    end

    ztrain = xtrain2*w;
    htrain = 1.0./(1.0 + exp(-ztrain));

    for j = 1:c+1
        somatoria = 0.0;
        for i = 1:mtrain
            somatoria = somatoria + alpha*double((htrain(i) - ytrain(i))*xtrain2(i, j));
        end
            w(j) = w(j) - somatoria;
    end
    Loss = 0.0;
    for i=1:train
        Loss = Loss + ytrain(i)*log(htrain(i)) + (1-ytrain(i))*log(1-ytrain(i));
    end
    Loss = Loss * (-1.0/mtrain);
    perda(cont) = Loss;
    cont = cont + 1;

    ytrainpred = double(htrain > 0.5);
    Acuracia_TrainingSet = mean(double(ytrainpred == ytrain)) * 100;
    epoch = epoch + 1;
end
fprint('Pasta_No:_%d', Pasta);

Acuracia_TrainingSet
figure
plot(perda);
```

```
histogram(htrain, 10);

mtest = lenght(ytest);
xtest2 = [ones(mtest, 1) xtest];
ztest = xtest2*w;

%Evaluate de model (Test Dataset)
htest = 1.0./(1.0 + exp(-ztest));
ytestpred = double(htest > 0.5);
ytestpred_sem = double(htest <= 0.5);

Acuracia_TestingSet = mean(double(ytestpred == ytest))*100
histogram(htest, 10);

Acuracia_Media = Acuracia_Media + Acuracia_TestingSet;

if Pasta == 1
        max = Acuracia_TestingSet
        PastaMax = Pasta;
else
        if Acuracua_TestingSet > max
                max = Acuracia_TestingSet;
                wMax = w;
                PastaMax = Pasta;
        end
end

%Matriz de Confusão
true_cancer = sum(double(ytest == 1))
true_no_cancer = sum(double(ytest == 0))
predicted_cancer = sum(double(ytestpred == 1))
true_positive = sum(double(ytest == 1) .*double(ytestpred == 1))
true_negative = sum(double(ytest == 0) .*double(ytestpred_sem == 1))
```

```
        false_cancer = sum(double(ytest == 0))

        Precision = 100.0*true_positive/predicted_cancer
        Recall = 100.0*true_positive/true_cancer

        specificity = 100.0*true_negative/true_no_cancer
        sensibility = 100.0*true_positive/true_cancer

        F1 = 2*Precision*Recall/(Precision + Recall)

        perda = 1;
        end

        PastaMax
        Acuracua_Media = Acuracia_Media/5
        max
xlswrite('w-cf-5fold.xls', wMax);
```

REFERÊNCIAS BIBLIOGRÁFICAS

A, Agresti. *Categorical Data Analysis*. 2a ed. New York: John Wiley Sons, 2002.

BARRETO, J. *Introdução às Redes Neurais Artificiais*. Laboratório de Conexionismo e Ciências Cognitivas -Universidade Federal de Santa Catarina, 2002. 17, 18.

BORRA; CIACCIO. Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods. Computacional Statistics and Data Analysis, 54:2976-2989.

BRAGA A.; CARVALHO, A. *Redes Neurais Artificiais: Teoria e Aplicações*. Rio de Janeiro: LTC, 2000. 237 p. 17, 24, 25“Câncer de Ovário,” 2021. Acessado em 06 de junho de 2021. [Online]. Disponível em: .
“Câncer de ovário é o mais agressivo dentre os tumores ginecológicos e requer atenção” 2021. Acessado em: 05 de junho de 2021. [Online]. Disponível em: http://www.fcecon.am.gov.br/cancer-de-ovario-e-o-mais-agressivo-dentre-os-tumores-ginecologicos-e-requer-atencao-alerta-fcecon/.

CARVALHO, P. C. et al. *Serum markers and mass spectrometry in the diagnosis of cancer*, Jornal Brasileiro de Patologia e Medicina Laboratorial, 2006, https://doi.org/10.1590/S1676-24442006000600005.

CARVALHO, P. C. et al. *Detection of potential serum molecular markers for Hodgkin's disease*. J Bras Patol Med Lab, v. 41, n. 3, p. 99-103, 2005.

CARVALHO, P. C. *Reconhecimento de padrões proteômicos e genômicos por aprendizagem de máquinas para o diagnóstico médico,* 2006. Tese (Mestrado) Departamento de Biologia Celular e Molecular do Instituto Oswaldo Cruz, Rio de Janeiro.

CHAND, H. S.; NESS, S. A.; KISIEL, W. *Identification of a novel human tissue factor splice variant that is upregulated in tumor cells*. Int J Cancer, v. 118, n. 7, p. 1713-20, 2005.

CONRADS, T. P. et al. *Cancer diagnosis using proteomic patterns*. Expert Rev Mol Diagn, v. 3, n. 4, p. 411-20, 2003.

D. Hosmer e S. Lemeshow. *Applied Logistic Regression*. 2a ed. New York: John Wiley Sons, 2000.

DIAMANDIS, E. P.; VAN DER MERWE, D. E. *Plasma protein profiling by mass spectrometry for cancer diagnosis: opportunities and limitations*. Clin Cancer Res, v. 11, n. 3, p. 963-5, 2005.

DOUGLAS, 2019. *ML – Métricas de Avaliação*. Acessado em: agosto de 2021. [Online]. Disponível em: https://medium.com/@douglasheberteempty/ml-m%C3%A9tricas-de-classifica%C3%A7%C3%A3o-97f57b27f61c.

ECHAN, L. A. et al. *Depletion of multiple high-abundance proteins improves protein profiling capacities of human serum and plasma*. Proteomics, v. 5, n. 13, p. 3292-303, 2005.“Estatística para câncer de ovário,” 2020. Acessado em: 09 de junho de 2021. [Online]. Disponível em: http://www.oncoguia.org.br/conteudo/estatistica-para-cancer-de-ovario/6045/228/.

FENN, J. B. et al. *Electrospray ionization for mass spectrometry of large biomolecules*. Science, v. 246, n. 4926, p. 64-71, 1989.

FLECK, L. *Redes Neurais Artificiais*: Princípios Básicos. Revista Eletrônica Científica Inovação e Tecnologia -Universidade Tecnológica Federal do Paraná, 2016. 16

FREITAS, G. A. L. *Aprendizagem Profunda Aplicada ao Futebol de Robôs*: Uso de Rede Neurais Convolucionais para Detecção de Objetos, Dissertação. Londrina, 2019.

FILIPOWICZ, W. et al. *Post-transcriptional gene silencing by siRNAs and miRNAs*. Curr Opin Struct Biol, v. 15, n. 3, p. 331-41, 2005.

Godbole S., Sarawagi S. (2004). *Discriminative Methods for Multi-labeled Classification*. In: Dai H., Srikant R., Zhang C. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2004. Lecture Notes in Computer Science, vol 3056. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24775-3_5.

GRONBORG, M. et al. *Biomarker discovery from pancreatic cancer secretome using a differential proteomics approach*. Mol Cell Proteomics, v. 5, p. 157-71, 2005.

GURNEY, K. *An Introduction to Neural Networks*. London: UCL Press, 1997. 316 p.17.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Ontario: Pearson Education, 1998. 818 p. 17, 19, 20, 21, 22, 23, 24, 25.

KAUFMANN, R. *Matrix-assisted laser desorption ionization (MALDI) mass spectrometry*: a novel analytical tool in molecular biology and biotechnology, Journal of Biotechnology, Volume 41, Issues 2–3, 1995, Pages 155-175, ISSN 0168-1656, https://doi.org/10.1016/0168-1656(95)00009-F.

KARAS M. et al. *Matrix-assisted ultraviolet laser desorption of non-volatile compounds. International Journal of Mass Spectrometry and Ion Processes*, v. 78, p. 53-68, 1987.

KHAN, S. *A Guide to Convolutional Neural Networks for Computer Vision*. Crawley: Morgan Claypool, 2018. 184 p. 16, 24, 28, 29, 30, 31, 32, 33.

KRUEGER-BECK, E. *Potencial de ação: do estímulo à adaptação neural*. Fisioter. Mov., Curitiba, v. 24, n. 3, p. 535-547, 2011. 18.

M. Douglas C. *Applied statistics and probability for engineers*. John Wiley Sons, Inc, 2002.

M. Hajmeera and I. *Basheerb, Comparison of logistic regression and neural network-based classifiers for bacterial growth in Food Microbiology*, Elsevier, 2003.

“Matplotlib: *Visualization with Python*,” 2021. Accessed on: May. 05, 2021. [Online]. Available: https://matplotlib.org/stable/index.html.

“Numpy: *About Us*,” 2020. Accessed on: May. 05, 2021. [Online]. Available: https://numpy.org/about/. Oncoguia, 2020. Maioria das brasileiras é diagnosticada com câncer de ovário já avançado. Acessado em: julho de 2021. [Online]. Disponível em: .Oncomed, 2020. Câncer de Ovário. Acessado em: julho de 2021.[Online]. Disponível em: https://www.oncomed.com.br/artigo-cancer-ovario.

P. Mesquita. *UM MODELO DE REGRESSAO LOGISTICA PARA AVALIACAO DOS PROGRAMAS DE POS-GRADUACAO˜NO BRASIL*. Dissertacao. JCAMPOS DOS GOYTACAZES, 2014.

S. Godbole and S. Sarawagi, *Discriminative methods for multi-labeled classification*, in Pacific-Asia conference on knowledge discovery and data mining. Springer, 2004.

S. Borra e A. Di Ciaccio. “*Measuring the prediction error. A comparison of cross-validation, bootstrap and co-variance penalty methods*”. Em: Computational Statistics Data Analysis 154 (2010), pp. 2976–2989. URL: https: www.sciencedirect.com / science / article / abs / pii / S0167947310001064. “Scikit-learn”. Shufflesplit. Acessado em: julho de 2021. [Online]. Disponível em: https://scikit-learn.org/.

”Tensorflow: *An end-to-end open source machine learning platform*,”2021. Accessed on: May. 07, 2021. [Online]. Available: https://www.tensorflow.org/.

”Tensorflow: *tf.keras.losses.binary crossentropy*,”2021. Accessed on: May. 07, 2021. [Online]. Available: https://www.tensorflow.org/api.

Adriana M. Tonini e Bráulio R.G.M. Couto, "Ensinando Geometria Analítica *com uso* do MATLAB," Departamento de Ciências Exatas e Tecnologia do Centro Universitário de Belo Horizonte / DECET - UniBH.

V. Gevert, A. Silva, F. Gevert e V. Ales. “*Modelos de Regressão Logística, Redes Neurais e Support Vector Machine (SVMs) na Análise de Crédito a Pessoas Jurídicas*”. Em: Revista Ciências Exatas e Naturais 12 (2010), pp. 270–293.

VESTAL, M. L. *Methods of ion generation*. Chem Rev, v. 101, n. 2, p. 361-75, 2001.

”Visualizing cross-validation behavior in scikitlearn”2021.