FACULDADE DE TECNOLOGIA DE SÃO PAULO – FATEC-SP
DEPARTAMENTO DE TECNOLOGIA DA INFORMAÇÃO – DTI
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS – ADS

TRABALHO DE CONCLUSÃO DE CURSO

USO DE *MACHINE LEARNING* NA PREVISÃO DE SAÚDE FETAL

São Paulo

Dezembro/2022

FACULDADE DE TECNOLOGIA DE SÃO PAULO – FATEC-SP
DEPARTAMENTO DE TECNOLOGIA DA INFORMAÇÃO – DTI
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS – ADS

TRABALHO DE CONCLUSÃO DE CURSO

USO DE *MACHINE LEARNING* NA PREVISÃO DE SAÚDE FETAL

Trabalho submetido como exigência parcial
para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas
Orientador: Prof. Dr. Silvio do Lago Pereira

São Paulo

Dezembro/2022

FACULDADE DE TECNOLOGIA DE SÃO PAULO – FATEC-SP
DEPARTAMENTO DE TECNOLOGIA DA INFORMAÇÃO – DTI
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS – ADS

Tiago de Souza Coviello

Uso de *Machine Learning* na previsão da saúde fetal

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador
Aprovado
_____

_____

_____

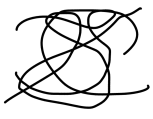
Conceito/Nota Final: 10,0

...

Orientador: Prof. Dr. Silvio do Lago Pereira

SÃO PAULO, 02 de 12 de 2022

Assinatura do orientador

Assinatura do aluno 

RESUMO

Apesar dos grandes avanços tecnológicos realizados nos últimos tempos, óbitos fetais e óbitos maternos, durante a gestação, ainda ocorrem de modo significativo.

Por isso, o objetivo do trabalho será analisar a aplicação da técnica de *Machine Learning* em dados obtidos pelo uso do método da cardiotocografia, visando verificar se essa tecnologia de inteligência artificial pode auxiliar na detecção do estado de saúde do feto para permitir que médicos tomem providências para prevenir a mortalidade da mãe e do filho em situações que indiquem risco.

Ademais, outros tópicos serão abordados, como a definição da ferramenta de *Machine Learning* e a comparação dos resultados obtidos com as pesquisas similares realizadas por outros pesquisadores.

Palavras-chaves*:* *Machine Learning*, Saúde fetal, Óbito fetal, Óbito materno.

ABSTRACT

Despite the big technological advances made in recent times, fetal deaths and maternal deaths, during pregnancy, still occur significantly.

Therefore, the objective of the work will be to analyze the application of the Machine Learning technique in data obtained using the cardiotocography method, aiming to verify if this artificial intelligence technique is able to help in the detection of the health status of the fetus to enable doctors to take measures to prevent mother and child mortality in situations that indicate risk.

Furthermore, other topics will be covered, such as the definition of the Machine Learning tool and the comparison of the results obtained with similar research done by other researchers.

Keywords: Machine Learning, Fetal health, Fetal death, Maternal death.

LISTA DE ABREVIAÇÕES

MMR - *Maternal Mortality Ratio* (Razão de Mortalidade Materna)
ReLU - *Rectified Linear Unit* (Unidade Linear Retificada)
MLP - *Multilayer Perceptron* (Perceptron com Múltiplas Camadas)
FHR - *Baseline Fetal Heart Rate* (Frequência Cardíaca Fetal da Linha de Base)

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1 - Introdução

1.1. Definição de óbito fetal, natimorto e óbito materno

O conceito de perdas fetais possui uma diversidade de definições relacionadas, devido à diversidade de parâmetros relacionados utilizados para definição, como idade gestacional (BRASIL, 2018, p. 361-362).

Portanto, para esse trabalho, será definido tanto o termo "natimorto" como o termo "óbito fetal", visando possibilitar o entendimento das pesquisas que serão mostradas. Ambos são conceitos similares às perdas fetais, e suas explicações serão passadas com base nas definições utilizadas nos estudos apresentados.

Para o termo "natimorto", será utilizada a definição usada por Unicef *et al*. (2020, p. 10) no relatório "*A Neglected Tragedy*", que foi padronizada como o nascimento de uma criança, sem sinais de vida, com 28 semanas ou mais de gravidez.

Já para o termo "óbito fetal", será empregado o significado utilizado no Brasil, descrito como:

> O Brasil adota a definição do óbito fetal como a morte do produto da gestação antes da expulsão ou de sua extração completa do corpo materno, independentemente da duração da gestação. A constatação do óbito é feita quando, após a separação do corpo da mãe, o feto além de não respirar, também não apresente nenhum outro sinal de vida, como batimentos do coração, pulsação do cordão umbilical ou movimentos efetivos dos músculos de contração voluntária. O aborto é a perda fetal com menos de 500 g e/ou comprimento menor que 25 cm, ou menos de 22 semanas de gestação (BRASIL, 2018, p. 362).

Por outro lado, para o termo "óbito materno", também chamado de “morte materna", será utilizada apenas uma definição, podendo ser apresentada como:

> Morte materna é a morte de uma mulher durante a gestação ou até 42 dias após o término da gestação, independentemente da duração ou da localização da gravidez. É causada por qualquer fator relacionado ou agravado pela gravidez ou por medidas tomadas em relação a ela. Não é considerada morte materna a que é provocada por fatores acidentais ou incidentais (BRASIL, 2007, p. 12).

1.2. Panorama atual

Nas últimas duas décadas, ocorreu uma diminuição de 21,4 natimortos por 1.000 nascimentos totais em 2000 para 13,9 natimortos por 1.000 nascimentos totais em 2019, indicando, dessa forma, uma redução de 35% no número de casos (UNICEF *et al*., 2020, p. 23).

Apesar disso, a ideia de que o problema seja pouco relevante, devido a essa melhoria, é errônea. Outros dados ainda apontam um cenário preocupante.

De acordo também com Unicef *et al*. (2020, p.3-6), a cada 16 segundos ocorre um caso de natimorto, totalizando quase 2 milhões de casos por ano. Os países de baixa e baixa-média renda concentram cerca de 84% desses casos. Nas condições atuais, um total de 20 milhões de casos de natimortos é estimado para essa década.

No Brasil, o número de mortes por óbito fetal, entre o período de 2013-2016, totalizou uma quantia de 127.330 casos, sendo uma média de 31.833 óbitos por ano (BRASIL, 2018, p. 361).

Dessa maneira, apesar da redução do número de casos, estes ainda representam uma quantia significativa, ainda mais quando se leva em conta a citação de Unicef *et al*. (2020, p.5), que afirmaram que a maioria dos casos podem ser evitados com medidas adequadas.

Outra problemática relacionada à questão de perdas fetais é a de óbitos maternos, que também possui um cenário similar.

Entre 2000 e 2017, a taxa de *Maternal Mortality Ratio* (MMR), que indica o número de mortes maternas por 100 mil nascimentos vivos, diminuiu cerca de 38% no mundo inteiro (ORGANIZAÇÃO MUNDIAL DA SAÚDE, 2019).

Contudo, também de acordo com a Organização Mundial da Saúde (2019), aproximadamente 810 mulheres morriam por dia, em 2017, por causas evitáveis relacionadas à gravidez e ao parto. A situação é intensificada em países de baixa e baixa-média renda, pois estes concentram cerca de 94% de todas as mortes maternas listadas.

Segundo Brasil (2020), no período de 2017-2018, no território brasileiro, também houve uma redução no número de casos, sendo esse percentual cerca de 8,4%. Entretanto, o país ainda obteve um MMR aproximado de 59, sendo um número de casos relevante.

Portanto, é visível que para ambas as problemáticas, é possível buscar alternativas para auxiliar na progressão da redução do número de casos, considerando que esses podem ser evitados.

1.3. *Machine Learning* (Aprendizado de Máquina)

Machine Learning é uma técnica que tem como objetivo estudar, arquitetar e melhorar modelos matemáticos que podem ser treinados com dados relacionados a um contexto para realizar predições sobre o futuro e tomar decisões, mesmo sem total conhecimento de todos os fatores externos (BONACCORSO, 2017, p.9).

No estudo, será utilizada a técnica para classificação de dados. De acordo com Aggarwal (2014, p. 2), o problema de classificação de dados é uma tentativa de apreender a relação entre um conjunto de variáveis que representam características e uma variável alvo de interesse.

Dessa forma, pode-se dizer que o modelo utiliza as características para prever o valor da variável alvo, com base no treinamento recebido pelo algoritmo.

Complementando a ideia, também pode-se citar Jadhav e Patil (2022, p.15), que afirmam que o propósito da tarefa é alocar uma classe para registros não categorizados.

Assim, com dados apropriados, a técnica pode auxiliar os profissionais a prevenir problemas durante a gravidez com indicação de possíveis casos de riscos que necessitam de medidas para preservar tanto a saúde do feto quanto a saúde da gestante.

Para obter os dados para a ferramenta, pode-se utilizar a tecnologia de cardiotocografia. De acordo com Furley (2012, p.2), essa é uma técnica de monitoração eletrônica para observar registros de informações como frequência cardíaca fetal, atividade uterina e movimentos fetais a fim de aferir o bem-estar do concepto.

Assim, a combinação das duas tecnologias pode ser uma solução viável para as problemáticas apresentadas. Ademais, ambas as tecnologias são fáceis de implementar e acessíveis, permitindo, desse modo, que sejam utilizadas também em países de baixa renda, os quais concentram a maior parte dos casos de óbitos.

Ademais, como poderá ser visualizado na próxima seção, a coleção de dados está rotulada em três diferentes classes, constituindo um cenário supervisionado para a tarefa de classificação.

Segundo Bonnacorso (2017, p.10-14), esse cenário de tarefa ocorre quando o conjunto de treinamento, ou seja, os dados passados para o algoritmo, é constituído tanto pela entrada de dados com as características dos registros quanto as saídas esperadas. Acrescentando a ideia, o autor também descreve o cenário de treinamento sem supervisão, quando o algoritmo tem que aprender como os dados devem ser agrupados de acordo com sua similaridade, ou seja, o algoritmo não possui as saídas esperadas e sim só as características, e o cenário de aprendizagem por reforço, em que o algoritmo aprende de acordo com a resposta proporcionada pelo ambiente, sendo uma informação mais qualitativa, como se uma ação é positiva ou não, e não uma medida precisa de erro.

A partir da visualização das diferenças entre os tipos de aprendizagem, é possível compreender melhor como a técnica de aprendizado de máquina irá ser aplicada nesse caso.

1.4. Metodologia de pesquisa

A metodologia de pesquisa será um estudo de caso realizado através do uso de uma base de dados. A coleção de dados foi disponibilizada por Campos *et al.* (2000, v. 9, n. 5, p. 311–318).

O conjunto de dados contém informações sobre 2126 cardiotocogramas, resultados advindos da técnica de cardiotocografia. Os registros foram classificados, por obstetras especialistas, em 3 classes de saúde, sendo elas: normal, suspeito e patológico.

Será aplicado a técnica de *Machine Learning* em cima dos dados do conjunto, utilizando dos algoritmos de *Random Forest* e *Artificial Neural Networks* na base de

dados, visando construir um modelo que classifique os registros de maneira correta. O desempenho do modelo será medido através de determinados parâmetros de medidas que serão explicados no Capítulo 5.

Outros pesquisadores também já realizaram pesquisas em cima da mesma base de dados, por isso também será realizada comparações com os seus estudos visando verificar se é possível obter resultados melhores ou se o limite já foi alcançado. Desse modo, será possível determinar se há algum modelo de *Machine Learning* definitivo para o caso de estudo ou se há vários modelos que podem ser viáveis.

Capítulo 2 - *Random Forest* (Floresta Aleatória)

2.1. *Decision Tree* (Árvore de Decisão)

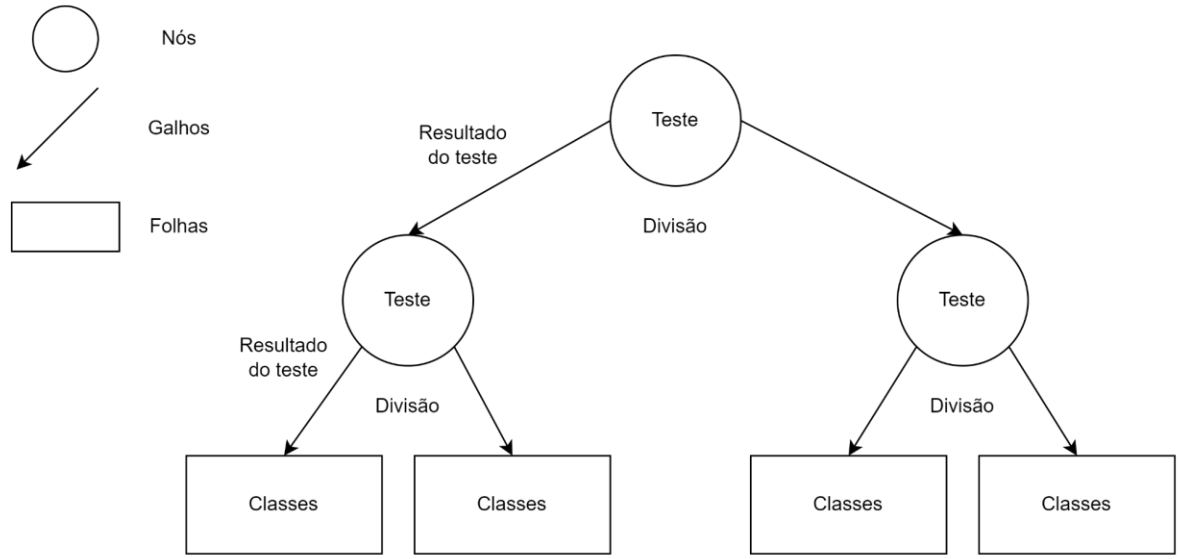
Um dos métodos de classificação mais utilizados é o algoritmo de Árvore de Decisão, que serve de base para o algoritmo de *Random Forest*. Assim, será passado uma explicação de seu funcionamento para possibilitar a definição do algoritmo *Random Forest*.

2.1.1. Conceituação

De modo geral, segundo Jadhav e Patil (2022, p.21), árvores de decisão são utilizadas para absorver informações valiosas, criando regras de decisão baseada nas informações passadas.

Ou seja, o algoritmo utiliza as variáveis de características passadas para montar suas regras de classificação.

Figura 1 - Árvore de Decisão



Fonte (Adaptado): KOZAK, 2018, p.4

Como pode ser visto na Figura 1, uma árvore de decisão é um gráfico em que todos os seus vértices são chamados de nós, arestas são chamadas de galhos e os vértices sem descendentes de folhas. A raiz representa o único vértice sem um nó pai (KOZAK, 2018, p.4).

Todos os nós contêm testes, em determinados atributos, gerados de acordo com o critério de separação selecionado. Assim, representam o modo para determinar

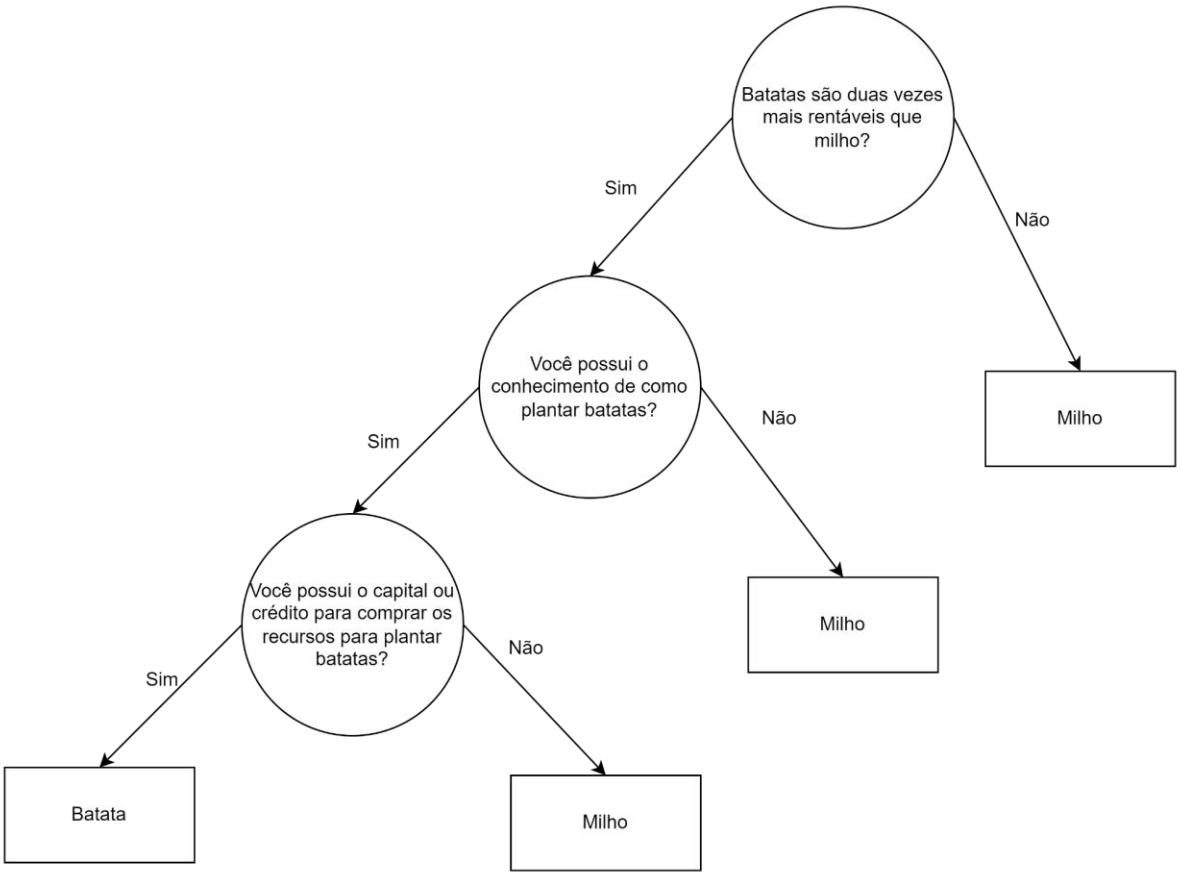
a divisão da base de dados, que realiza a separação de acordo com os atributos dos registros (KOZAK, 2018, p.4).

Os resultados dos testes são representados pelos galhos (KOZAK, 2018, p.4).

2.1.2. Exemplificação

Visando exemplificar um teste de árvore de decisão de classificação, pode-se usar o exemplo dado por Gladwin (1989, p.14), que supôs um cenário de um fazendeiro, com possivelmente pouco capital, que tenha que realizar uma decisão de plantar batatas, colheita voltada para ganhos financeiros, ou milho, colheita voltada para subsistência, como pode ser visto na Figura 2.

Figura 2 - Exemplificação de Árvore de Decisão



Fonte (Adaptado): GLADWIN, 1989

O primeiro teste realizado na árvore de decisão da Figura 2 é "Batatas são duas vezes mais rentáveis que milho?", visando verificar se realmente é justificável a motivação de plantar batatas para ganho. Caso a resposta do teste seja não, então não é justificável a plantação de batatas, mas caso seja sim, então deve-se realizar o segundo teste.

O segundo teste é "Você possui o conhecimento de como plantar batatas?", visando verificar se é possível realizar o cultivo com o conhecimento atual do fazendeiro, pois caso ele não possua o conhecimento, ele terá que obter, indicando possíveis custos elevados com aprendizado, ou ele terá muitos desperdícios na plantação. Caso a resposta do teste seja não, então também não se justifica a plantação de batatas, mas caso seja sim, deve-se realizar um último teste.

O teste final é "Você possui o capital ou crédito para comprar os recursos para plantar batatas?", visando verificar se é economicamente viável o plano de plantio. Caso a resposta seja não, então não se deve plantar batatas, mas caso seja sim, então é possível realizar a plantação.

Nesse exemplo, a variável alvo pode receber tanto o valor de classe “Batata” quanto o valor de classe “Milho”.

2.1.3. Critérios

Existem diversos tipos de critérios para construção de árvores de decisão.

Esses critérios servem para selecionar os atributos para os nódulos raiz em cada nível da árvore de decisão, como descreve Jadhav e Patil (2022, p.23).

Os critérios que serão descritos são o critério de Entropia e o critério de Índice-Gini, os quais são os mais pertinentes ao estudo e aos testes. Ambos os critérios possuem um processo similar, com diferenças nos cálculos realizados.

2.1.3.1. Entropia

Nesse critério, o algoritmo procura o atributo com maior ganho de informação para colocá-lo como nódulo raiz da árvore de decisão, pois o atributo com maior ganho de informação divide mais facilmente os dados. Depois de selecionar o nódulo raiz, o algoritmo procura o segundo atributo com maior ganho de informação e o coloca como nódulo no segundo nível. O processo é repetido até *n-1* atributos serem abrangidos (SINGH; CHHABRA, 2021, p.33).

Figura 3 - Equação de Ganho de Informação

$$IG(S, X) = \mathrm{E}(\mathrm{S}) - \mathrm{E}(\mathrm{S}|\mathrm{X})$$

Fonte: SINGH; CHHABRA, 2021, p.33

A Figura 3 representa a equação do ganho de informação. Sendo *E(S)* a entropia do conjunto de dados de amostra e *E(S|X)* a soma das entropias após dividir o conjunto de dados em *m* diferentes classes baseadas em um determinado atributo *X* (SINGH; CHHABRA, 2021, p.33).

Figura 4 - Equação de Entropia

$$E(S) = \sum_{i=1}^{n} -(p_i log_2 p_i)$$

Fonte: SINGH; CHHABRA, 2021, p.33

A Figura 4, por sua vez, representa o cálculo do valor da entropia, sendo p_i a probabilidade da i-ésima classe e *E(S)* é a entropia de um conjunto de dados de amostra *S* (SINGH; CHHABRA, 2021, p. 33).

Logo, o processo, de forma geral, consiste em selecionar os atributos com maior capacidade de dividir os dados para cada nível da árvore, utilizando das equações de Ganho de Informação e Entropia no procedimento.

2.1.3.2. Índice de Gini

Como descreve Rebala, Ravi e Churiwala (2019, p.86-87), o critério de Gini é uma medida de quanto frequentemente um elemento aleatoriamente escolhido de um conjunto de dados é incorretamente classificado se for classificado de acordo com a distribuição de categorias no conjunto. Desse modo, altos índices de impureza de Gini descrevem uma chance grande de classificação errada e baixos índices de impureza descrevem uma chance menor de classificação errada. O objetivo na construção da Árvore de Decisão é uma divisão com o menor índice de impureza de Gini para os nódulos filhos.

O índice de Gini é calculado para todos n - 1 atributos, variando seu valor de 0 a 1. O atributo com menor valor de Gini é selecionado como nódulo raiz da árvore de decisão. O processo é repetido pelo algoritmo até todos os n-1 atributos serem adicionados à árvore de decisão (SINGH; CHHABRA, 2021, p.33).

Figura 5 - Equação de Gini

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Fonte: SINGH; CHHABRA, 2021, p.34

Na Figura 5, pode ser visualizada a equação utilizada para calcular o índice de Gini, sendo que p_i representa a probabilidade de um objeto ser classificado em uma determinada classe (SINGH; CHHABRA, 2021, p.34).

Como pode ser visto, o processo consiste na seleção de atributos para cada nódulo de raiz, os quais são escolhidos baseados na sua capacidade de classificação dos elementos, utilizando a fórmula de Gini para esse selecionamento. O processo se assemelha, em certo grau, ao critério de Entropia.

2.2. *Random Forest*

O algoritmo de *Random Forest* funciona com base na construção de um conjunto de árvores de decisão, em que cada uma possui um conjunto aleatório escolhido de atributos. Apenas um número predefinido de características é escolhido aleatoriamente para cada árvore de decisão (MOOLAYIL, 2016, p.208).

Cada árvore de decisão de classificação individual do conjunto de árvores do algoritmo de *Random Forest* é usada para prever a classe dos dados de entrada. A classe com mais votos torna-se a classe resultante do modelo (SINGH; CHHABRA, 2021, p.34).

A adição de aleatoriedade nos atributos de cada árvore ajuda o algoritmo de *Random Forest* a possuir mais estabilidade que o algoritmo de Árvore de Decisão (MOOLAYIL, 2016, p.208).

Acrescentando a ideia, de acordo com Panesar (2019, p. 136), o propósito do algoritmo de florestas aleatórias é prevenir o problema de *overfitting*.

Visando deixar claro o que é *overfitting*, pode-se usar a definição dada por Berman (2013, p.148), que explicou que esse problema ocorre quando uma fórmula descreve um conjunto de dados de maneira precisa, mas não consegue predizer corretamente outros conjuntos de dados, ou seja, a fórmula enfrenta problemas em

descrever características de comportamento do sistema, prejudicando, assim, assim a capacidade de generalização do algoritmo. É um problema que também ocorre com outros algoritmos de *Machine Learning*.

Cabe ressaltar que o algoritmo possui um tempo de predição maior que o algoritmo de Árvore de Decisão, porque um grupo de modelos de árvores de decisão deve funcionar em cima de um mesmo registro para realizar a predição (SINGH; CHHABRA, 2021, p.34).

Devido à construção de várias árvores de decisão, o algoritmo obtém vantagens em seu modo de funcionamento, se adequando a diversas situações de classificação. Dessa forma, o algoritmo pode proporcionar bons resultados para o experimento.

Capítulo 3 - *Artificial Neural Network* (Rede Neural Artificial)

3.1. Conceituação

Rede Neural Artificial é uma técnica de *Machine Learning* que simula o mecanismo de aprendizagem dos organismos biológicos (AGGARWAL, 2018, p.1).

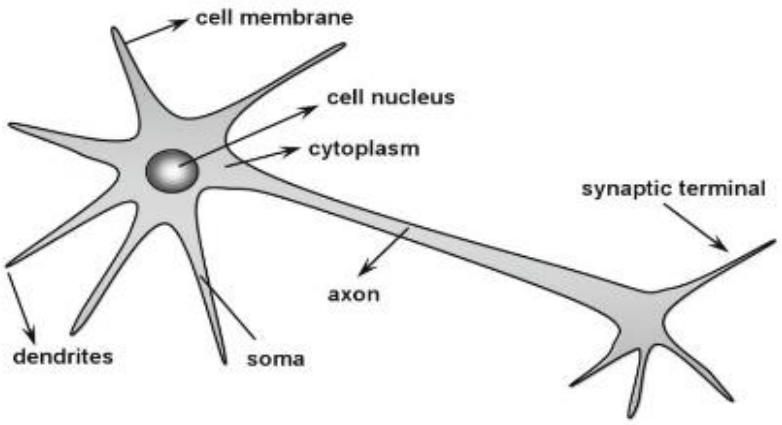
O sistema humano contém células, que são referidas como neurônios. Os neurônios são conectados a outros com uso dos axônios e dendritos, e a região de conexão entre axônios e dendritos é denominada de sinapse. A força das conexões sinápticas frequentemente muda em resposta aos estímulos externos, sendo essa mudança o meio para a aprendizagem nos organismos vivos (AGGARWAL, 2018, p.1).

Os axônios são responsáveis por conduzir os impulsos elétricos para outros neurônios de conexão ou para neurônios diretamente conectados com o tecido muscular (neurônios eferentes). Os dendritos, por sua vez, são os responsáveis por adquirir continuamente estímulos de outros neurônios ou do ambiente externo (SILVA *et al*., 2016, p.9).

As sinapses são conexões que possibilitam a transmissão de impulsos elétricos do axônio de um neurônio para os dendritos de outro neurônio (SILVA *et al*., 2016, p.9).

Esse mecanismo biológico é simulado em redes neurais artificiais, que contém unidades de computação referidas como neurônios (AGGARWAL, 2018, p.1).

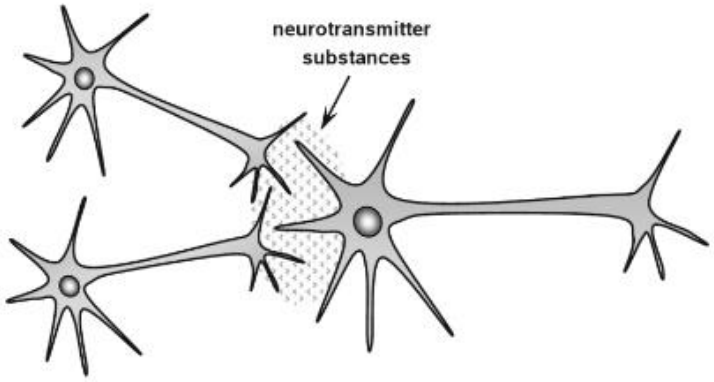
Figura 6 - Neurônio Biológico



Fonte: SILVA *et al*., 2016, p.9

Na Figura 6, pode ser visualizado um Neurônio Biológico. É possível visualizar tanto os dendritos, referidos como “*dendrites*”, quanto o axônio, referido como “*axon*”, ambos componentes citados anteriormente.

Figura 7 - Sinapse

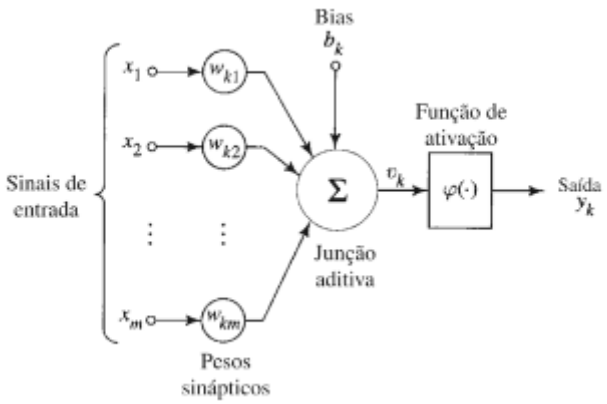


Fonte: SILVA *et al*., 2016, p.10

Já na Figura 7, é possível observar a realização de uma sinapse entre neurônios, sendo possível verificar a neurotransmissão de substâncias.

3.2. Neurônio Artificial

Figura 8 - Neurônio Artificial



Fonte: HAYKIN, 2007, p.36

Na ilustração apresentada pela Figura 8, é possível visualizar a estrutura de um neurônio artificial.

Três elementos básicos podem ser identificados: Um conjunto de sinapses, um somador e uma função de ativação (HAYKIN, 2007, p.36-37).

No primeiro elemento, o conjunto de sinapses, especificamente, é possível ver um sinal x_i, sendo multiplicado pelo peso sináptico w_{k1}, na entrada da sinapse j

conectada ao neurônio k. Os índices do peso sináptico representam, respectivamente, o neurônio em questão e o terminal de entrada da sinapse à qual o peso se refere. O peso sináptico de um neurônio artificial pode estar em um intervalo que também inclua valores negativos, além dos valores positivos, diferentemente de uma sinapse cerebral (HAYKIN, 2007, p.36).

O segundo elemento, o somador, é responsável por somar os sinais de entradas, que foram ponderados pelas respectivas sinapses dos neurônios (HAYKIN, 2007, p.37).

O terceiro elemento, a função de ativação, tem como função restringir a amplitude de saída de um neurônio, limitando o intervalo de amplitude do sinal de saída a um determinado valor finito (HAYKIN, 2007, p.37).

Também há um viés b_k, que tem como efeito aumentar ou diminuir a entrada líquida da função, dependendo de seu sinal (HAYKIN, 2007, p.37).

O viés atua em conjunto com o somador para produzir o potencial de ativação u_k, que é o resultado produzido pela diferença entre os dois. Se o seu valor positivo, ou maior ou igual ao viés, será produzido um potencial excitante, caso contrário, será produzido um potencial inibitório (SILVA *et al*., 2016, p.12).

Por fim, a saída γ_k é o valor final produzido pelo neurônio depois de receber as entradas, e pode ser utilizado como entrada para outros neurônios conectados sequencialmente (SILVA *et al*., 2016, p.12).

De modo geral, o processo pode ser resumido, como descrito por Silva *et al*. (2016, p.13), em recebimento dos valores de entrada, cálculo da multiplicação de cada entrada pelo peso correspondente, obtenção do potencial de ativação pela soma ponderada das entradas e subtração do viés, aplicação da função de ativação visando limitar a saída do neurônio e compilação da saída por meio do emprego da função de ativação neural no potencial de ativação.

Portanto, o neurônio artificial é responsável por receber entradas de valores e realizar ajustes nesses valores. Esse processo, por sua vez, possui similaridades com o mecanismo existente em organismos biológicos, devido ao processo ser baseado nesse mecanismo.

3.3. Funções de Ativação

Como explicado na seção anterior, uma função de ativação é responsável por delimitar a amplitude do valor de saída a um determinado valor limitado.

Existem diversas funções de ativação, mas assim como nos critérios de árvores de decisão, serão apenas apresentadas as mais pertinentes ao estudo e realização dos testes. Sendo assim, 5 funções foram selecionadas, sendo elas: Função de Limiar, Função Identidade, Função Logística, Função Hiperbólica e Função *Rectified Linear Unit* (ReLU).

3.3.1. Função de Limiar

A função também é chamada de "*Step Function*" e "Função de Heavside".

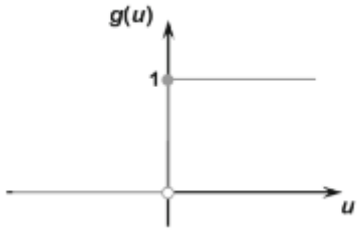
O resultado da função assumirá valor unitário quando o potencial de ativação do neurônio for maior ou igual a zero ou valor nulo em caso contrário (SILVA *et al*., 2016, p.13).

Figura 9 - Função de Limiar

$$g(u) = \begin{cases} 1, & \text{if } u \geq 0 \\ 0, & \text{if } u < 0 \end{cases}$$

Fonte: SILVA *et al*., 2016, p.13

Figura 10 - Gráfico da Função de Limiar



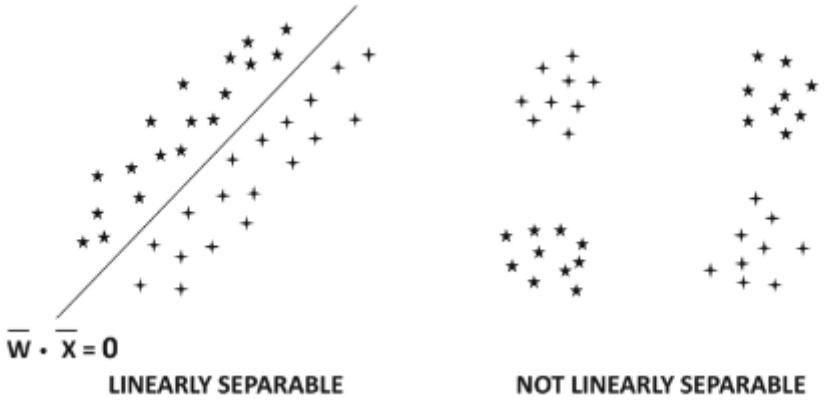
Fonte: SILVA *et al*., 2016, p.14

A Figura 9 e a Figura 10, representam, respectivamente, a função de limiar e o gráfico desta função.

3.3.2. Função Identidade

Também é chamada de função linear.

Figura 11 - Diferença entre Função Linear e Não Linear



Fonte: AGGARWAL, 2018, p.8

Como pode ser visto na Figura 11, existe uma diferença entre funções lineares e funções não lineares. Funções lineares, à esquerda da Figura 11, podem ser separadas por uma reta, diferentemente das funções não linearmente separáveis, à direita da Figura 11.

Todas as próximas funções de ativação que serão apresentadas são exemplos de funções não linearmente separáveis.

A função identidade produz uma saída com resultado igual ao potencial de ativação u (SILVA *et al*., 2016, p.18).

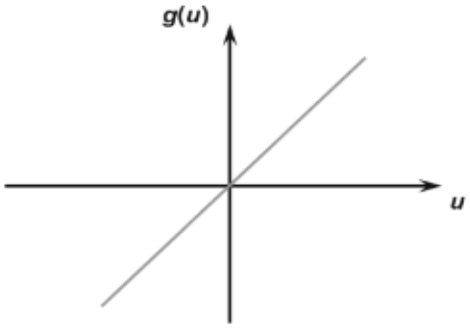
Figura 12 - Função Identidade

$$g(u) = u$$

Fonte: SILVA *et al*., 2016, p.18

Na equação apresentada pela Figura 12, pode ser visualizada a função identidade.

Figura 13 - Gráfico da Função Identidade



Fonte: SILVA *et al*., 2016, p.18

Já na Figura 13, pode ser visualizado o gráfico que representa a função identidade.

3.3.3. Função Logística

A função logística sempre produz valores de saída entre zero e um (SILVA *et al*., 2016, p.15).

Na equação apresentada pela Figura 14, é possível visualizar a representação dessa função.

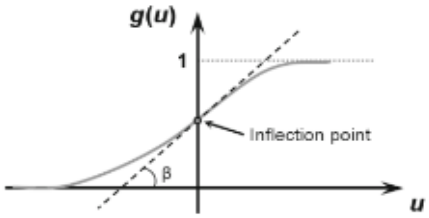
Figura 14 - Função Logística

$$g(u) = \frac{1}{1 + e^{-\beta \cdot u}} \ ,$$

Fonte: SILVA *et al*., 2016, p.15

β representa uma constante real associada com a inclinação da função no seu ponto de inflexão (SILVA *et al*., 2016, p.15).

Figura 15 - Gráfico da Função Logística



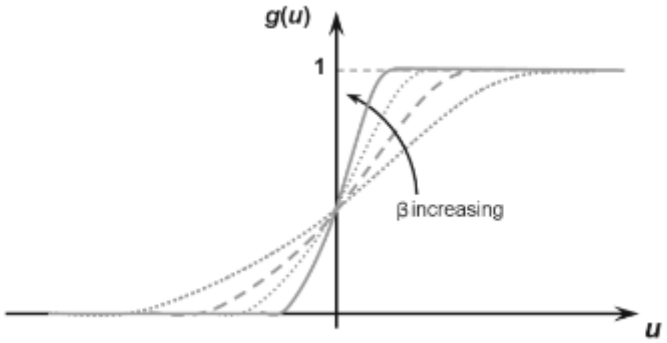
Fonte: SILVA *et al*., 2016, p.16

Na Figura 15, é possível observar a representação gráfica da função logística, incluindo seu ponto de inflexão, nomeado como “*Inflection point*”.

Por ser um tipo de função sigmóide, possui gráfico em formato de “S” e seus valores variam em um intervalo contínuo de 0 e 1, diferentemente da função de limiar que possuía o valor de 0 ou 1. (HAYKIN, 2007, p.40)

Deixando de maneira mais clara a definição de função sigmóide, essa é designada como uma função estritamente crescente que exibe um balanceamento adequado entre comportamento linear e não-linear (HAYKIN, 2007, p.40).

Figura 16 - Influência de β na Função Logística



Fonte: SILVA *et al*., 2016, p.16

Na Figura 16, é possível visualizar a ilustração de um gráfico, representando a influência de β na função logística.

É possível observar também na Figura 16, a semelhança com a função de limiar, quando β tende ao infinito (SILVA *et al*., 2016, p.15).

A função sofre do problema de desaparecimento do gradiente, que ocorre quando os valores de x estão distantes da origem e quando há várias camadas com ativação sigmóide. Nessas circunstâncias, o valor do gradiente começa a ficar muito pequeno, próximo de zero, tornando as mudanças de peso muito pequenas ou até possível de serem desconsideradas. Assim, a rede neural artificial fica presa na configuração atual de parâmetros e não consegue aprender mais (AGHDAM; HERAVI, 2017, p.71-72).

Complementando a ideia do gradiente, segundo Ravichandiran (2019, p.23), o gradiente, com sua técnica de descida do gradiente, permite que a rede neural aprenda os valores ideais para os pesos iniciados aleatoriamente. Com os valores ideais, a rede neural artificial consegue prever a saída correta e minimiza a perda, ou erro.

Assim, a ideia do desaparecimento do gradiente é que a rede neural, em determinado momento, não consegue mais evoluir e diminuir seu erro.

O gradiente também será explicado novamente na seção de *backpropagation*, para relacionar a importância da técnica com a etapa de aprendizado e ajustes de pesos.

3.3.4. Função Hiperbólica

Diferentemente do caso anterior, segundo Silva et al. (2016, p.16), os resultados dessa função sempre assumiram valores entre -1 e +1.

Na Figura 17, pode ser vista a função Hiperbólica.

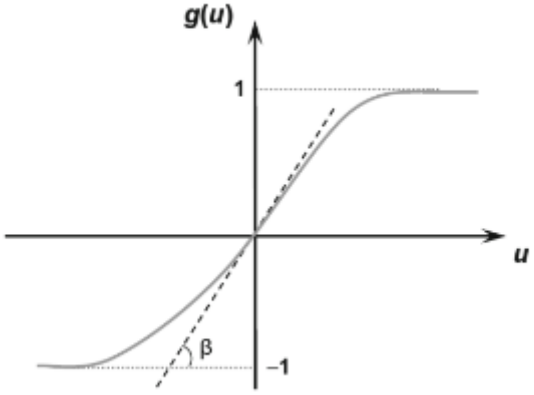
Figura 17 - Função Hiperbólica

$$g(u) = \frac{1 - e^{-\beta \cdot u}}{1 + e^{-\beta \cdot u}} \ ,$$

Fonte: SILVA *et al*., 2016, p.16

β, nesse caso, também representa uma constante real associada com a inclinação da função no seu ponto de inflexão (SILVA *et al*., 2016, p.16).

Figura 18 - Gráfico da Função Hiperbólica

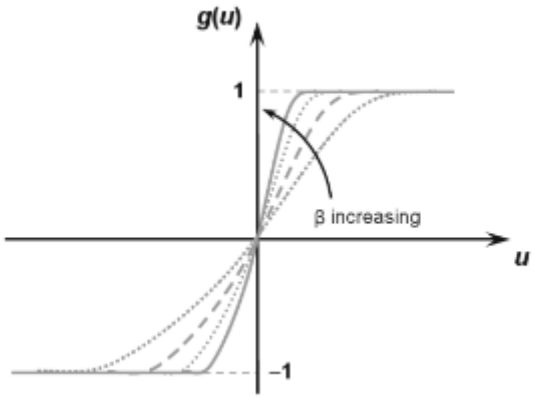


Fonte: SILVA *et al*., 2016, p.16

A Figura 18, por sua vez, apresenta o gráfico que representa a função hiperbólica. Como pode ser visto nessa figura, o gráfico da função hiperbólica possui similaridades com o gráfico da função logística.

SILVA *et al*. (2016, p.17) explica que ambas são funções sigmóides, então a similaridade ocorre devido a esse fator.

Figura 19 - Influência de β na Função Logística



Fonte: SILVA *et al*., 2016, p.17

A Figura 19 apresenta um gráfico que permite constatar que quanto maior o valor de β, maior a inclinação da função, assim como ocorre no caso da função logística (SILVA *et al*., 2016, p.17).

Também similar a função logística, a função sofre o problema da perda do gradiente (AGHDAM; HERAVI, 2017, p.72).

3.3.5. Função *Rectified Linear Unit* (ReLU)

A função ReLu possui alcance de [0,∞) e é uma função de ativação muito simples e eficiente que é utilizada em Redes Neurais Artificiais com muitas camadas ocultas, pois funciona melhor que as funções anteriormente apresentadas, já que não possui o problema de desaparecimento do gradiente, sempre produzindo um gradiente forte nas regiões onde ocorria o desaparecimento do gradiente para as outras funções (AGHDAM; HERAVI, 2017, p.73-74).

Figura 20 - Função ReLU

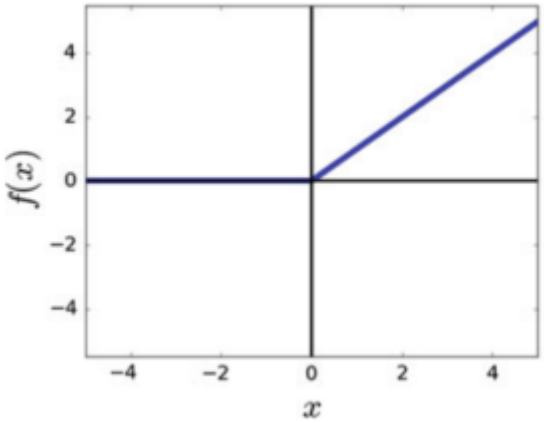
$$\mathscr{G}_{relu}(x) = \max(0, x)$$

$$\mathscr{G}'_{relu}(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

Fonte: AGHDAM; HERAVI, 2017, p.74

A definição da função ReLU é demonstrada na Figura 20.

Figura 21 - Gráfico da Função ReLU



Fonte: AGHDAM; HERAVI, 2017, p.74

A Figura 21, por sua vez, apresenta uma ilustração de um gráfico que representa a função ReLU.

3.4. Arquitetura de Redes Neurais Artificiais

Nesta seção, será abordada a arquitetura de redes neurais artificiais, como seus componentes e a forma como a estrutura pode ser formada.

3.4.1. Componentes

A estrutura geral da rede neural artificial pode ser dividida em três camadas diferentes (SILVA *et al*., 2016, p.21).

A primeira é a camada de entrada, responsável por receber informações do ambiente externo, como dados, sinais e características (SILVA *et al*., 2016, p.21).

A segunda parte de camadas é nomeada de “camadas ocultas”, também chamadas de “camadas invisíveis" ou “intermediárias”, que são camadas com neurônios responsáveis por extrair padrões das informações passadas. Essas camadas executam a maior parte do processamento interno da rede neural (SILVA *et al*., 2016, p.22).

A terceira camada é a camada de saída, também composta de neurônios, com responsabilidade de produzir o resultado da rede neural. Essa saída produzida é resultante do processamento realizado pelos neurônios nas camadas anteriores (SILVA *et al*., 2016, p.22).

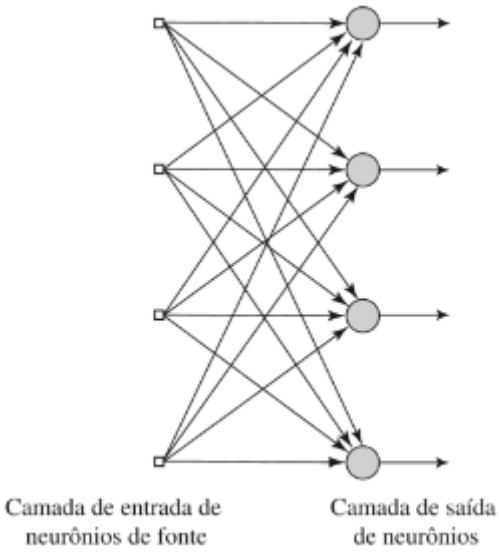
3.4.2. Redes Neurais Artificiais de Camada Única

É composta por uma camada de entrada de nós de fonte e uma camada de saída de neurônios (nós computacionais) (HAYKIN, 2007, p.46).

É chamada também de "rede de camada única", se referindo à camada de saída, sendo que a camada de entrada não é incluída na contagem por não realizar nenhum tipo de computação (HAYKIN, 2007, p.47).

O fluxo de informação é unidirecional, fluindo apenas na direção da camada de entrada para a camada de saída (SILVA *et al*., 2016, p.22).

Figura 22 - Rede Neural Artificial de Camada Única



Fonte: HAYKIN, 2007, p.47

Como pode ser visto na Figura 22, esse tipo de estrutura não possui nenhuma camada oculta. Assim, a estrutura é utilizada, de modo geral, para tarefas mais simples e que não necessitem de grande processamento.

Segundo Silva *et al*. (2016, p.22), os usos mais comuns desse tipo de estrutura são em classificações de padrões e problemas de filtragem linear.

3.4.3. Redes Neurais Artificiais com Múltiplas Camadas

Diferentemente do tipo de arquitetura anterior, esse tipo de estrutura é composto por um ou mais camadas ocultas (SILVA *et al*., 2016, p.23).

Os nós de fonte da camada de entrada da rede neural fornecem o vetor de entrada que constitui os sinais de entradas aplicados aos neurônios (nós

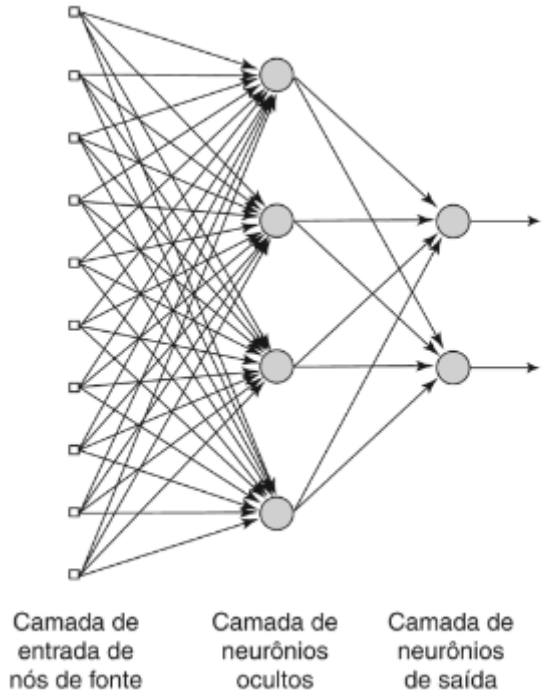
computacionais) localizados na segunda camada (isto é, a primeira camada oculta). Os sinais de saída provenientes dessa segunda camada servem de entrada para a terceira camada e assim funciona até o final da rede (HAYKIN, 2007, p.47).

Dessa forma, o tipo de estrutura de multicamadas é também chamado de "*feed-forward networks*", que significa sucessivas camadas que alimentam as camadas localizadas à frente, indo da direção da entrada até a saída (AGGARWAL, 2018, p.17).

Muitas vezes essas redes neurais também podem ser chamadas de "*Multilayer Perceptron"* (MLP) (HAYKIN, 2007, p.183).

Os neurônios das camadas ocultas intervêm entre a entrada externa e a saída da rede de uma maneira benéfica, permitindo, assim, que a rede neural artificial consiga extrair estatísticas de alto valor. Quando o tamanho da camada de entrada é grande, essa capacidade é muito importante (HAYKIN, 2007, p.47).

Figura 23 - Rede Neural Artificial com Múltiplas Camadas



Fonte: HAYKIN, 2007, p.48

Na Figura 23, pode ser visualizada uma rede neural artificial com múltiplas camadas, possuindo uma camada oculta. Deve-se notar que a camada de neurônios de saída, assim como na rede neural artificial de camada única, gera a resposta final

do algoritmo. Ademais, o modelo pode possuir quantas camadas ocultas forem necessárias.

A camada oculta pode ser configurada em diversas implementações do algoritmo de *Artificial Neural Network*. No Capítulo 4, será visto que a quantidade de camadas ocultas e seus respectivos números de neurônio são um dos hiperparâmetros a serem configurados.

De acordo com Silva *et al*. (2016, p.23), esse tipo de estrutura pode ser empregue em diversos tipos de tarefas, como problemas de classificação de padrões, identificação do sistema, controle de processos e otimização.

3.5. Aprendizagem

Compreender o processo de aprendizagem das redes neurais artificiais é necessário para conseguir entender a próxima seção, que abordará a técnica de *backpropagation*, responsável pelo processo de ajuste da rede neural para aumento de sua acuracidade.

Dessa forma, será apresentado brevemente a regra de aprendizagem por correção de erros, que é a regra mais pertinente a técnica.

De modo geral, resumindo o processo descrito por Haykin (2007, p.76-78), o sinal de saída da rede neural é comparado com uma resposta desejada, ou também chamada de saída alvo, e a diferença é chamada de sinal de erro. Esse sinal aciona um mecanismo de controle que tem como missão aplicar uma sequência de ajustes corretivos aos pesos sinápticos do neurônio, visando aproximar o sinal de saída da resposta desejada. Os ajustes continuam até o sistema atingir um estado estável, com os pesos sinápticos essencialmente estabilizados.

Vale ressaltar que os ajustes podem ser também finalizados devido ao limite pré-determinado ser atingido. Esse limite, em muitos dos algoritmos, pode ser determinado pelo usuário.

Os ajustes citados constituem o processo de *backpropagation*, que será explicado na próxima seção.

3.6. *Backpropagation* (Retropropagação)

Em redes neurais artificiais de camada única, o processo de treinamento é relativamente orientado para frente porque o erro pode ser computado como uma função direta dos pesos, que permite uma fácil computação do gradiente (AGGARWAL, 2018, p.21).

Em redes neurais artificiais com múltiplas camadas, o erro é uma complicada função composta dos pesos das camadas iniciais. Dessa forma, para computar o gradiente de uma função composta, é necessário o algoritmo de *backpropagation* (AGGARWAL, 2018, p.21).

Como uma introdução geral, pode-se dizer que o algoritmo é uma sequência de cálculos utilizados para ajuste dos pesos sinápticos da rede neural artificial visando diminuir o erro, ocorrendo após a execução da técnica de *feedforward*.

O algoritmo de *backpropagation* tem como objetivo minimizar o erro ao quadrado entre a saída atual e a saída desejada de uma rede neural artificial sobre uma coleção de dados (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.3-4).

Conforme pode ser notado, o cálculo de erro segue a descrição apresentada na seção anterior.

Figura 24 - Cálculo do Erro

$$E_p = \frac{1}{2}\sum_k (d_k - y_k)^2$$

Fonte: SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.4

Na Figura 24, é possível visualizar a fórmula utilizada para o cálculo do erro.

E_p representa o erro para a amostra p, y_k e d_k são, respectivamente, a saída real e a saída desejada do neurônio k na camada de saída, sendo que k percorre sobre todos os neurônios de saída (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.4).

O algoritmo possui duas fases conhecidas nomeadas como "*forward pass*" e "*backward pass*" (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.4).

Em tradução livre, "*forward pass*” pode ser chamada de passagem para frente e “*backward pass*” pode ser chamada de passagem para trás.

3.6.1. *Forward pass*

Na fase de *forward pass*, ocorre o cálculo da ativação dos neurônios de cada camada, indo da primeira camada oculta até a camada de saída (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.4).

As entradas são passadas para a rede neural artificial nessa etapa, resultando em diversas computações através das camadas, usando o atual conjunto de pesos. A saída final pode ser comparada com a instância de treinamento e a derivada da função de perda relativa à saída é computada (AGGARWAL, 2018, p.21).

A explicação de *feed-forward*, chamada de “*forward pass*” nesse caso, foi também apresentada brevemente na descrição de Redes Neurais Artificiais com Múltiplas Camadas.

Figura 25 - Ativação do Neurônio

$$a_k = f(\sum_i w_{ik} a_i)$$

Fonte: SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.4

A Figura 25 apresenta o cálculo de ativação de um neurônio. No exemplo utilizado pelo autor, a função f(x) utilizada é uma função sigmóide similar a função logística, anteriormente explicada.

O elemento a_k representa a ativação do neurônio k, w_{ik} é o peso sináptico da conexão entre o neurônio i e o neurônio k. a_i são as ativações de neurônios na camada i (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.4).

3.6.2. *Backward pass*

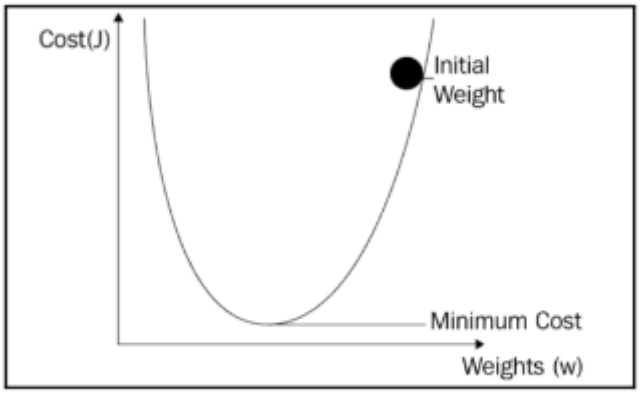
3.6.2.1. Definição

Na fase de *backward pass*, ocorre o cálculo da parcial derivada δ_k do erro associado com cada neurônio k (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.5).

O principal objetivo da fase é aprender o gradiente da função de perda com

relação aos diferentes pesos. Os gradientes descobertos serão utilizados para atualizar os pesos sinápticos (AGGARWAL, 2018, p.21).

Figura 26 - Descida do Gradiente



Fonte: RAVICHANDIRAN, 2019, p.24

Como pode ser visto na Figura 26, há um gráfico que relaciona o custo, no eixo vertical, com os pesos, no eixo horizontal. Além disso, o gráfico também representa os pesos iniciais, que como abordado anteriormente, são escolhidos aleatoriamente. O propósito da Figura 26 é representar como funciona a metodologia de descida do gradiente.

Deve-se minimizar a função de custo, por meio da chegada ao ponto mais baixo onde o custo é mínimo. Desse modo, é utilizado o cálculo do gradiente, que possibilita a movimentação de um ponto para outro, possibilitando, assim, mover o peso inicial para o ponto onde o custo é mínimo (RAVICHANDIRAN, 2019, p.24-25).

3.6.2.2. Realização dos cálculos

Após a ideia geral da técnica de *backward pass* ter sido definida, será apresentado, brevemente, os cálculos utilizados para o método.

Figura 27 - Cálculo de Derivada para Camada de Saída

$$\delta_k = (d_k - y_k)\, y_k\, (1 - y_k).$$

Fonte: SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.5

Figura 28 - Cálculo de Derivada para Camada Oculta

$$\delta_k = y_k\, (1\ -\ y_k) \sum_l \delta_l w_{kl}$$

Fonte: SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.5

As Figura 27 e Figura 28 apresentam, respectivamente, a equação da derivada, tanto para a camada de saída quanto para a camada oculta.

O componente ⌊ é responsável por percorrer todos os neurônios da próxima camada (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.5).

Figura 29 - Cálculo de Incremento dos Pesos Sinápticos

$$\Delta w_{kl} = \alpha \ \delta_l \ y_k$$

Fonte: SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.5

A Figura 29 demonstra como é realizado o cálculo de incremento dos pesos.

O cálculo ocorre após a computação das derivadas. α representa a taxa de aprendizado (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.5).

Figura 30 - Cálculo de Atualização do Peso Sináptico

$$w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n) + \beta \Delta w_{kl}(n-1)$$

Fonte: SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.5

A Figura 30, por sua vez, mostra como é realizado o cálculo para atualização dos pesos sinápticos.

O componente n representa a iteração atual e β representa a taxa de momento (SARATCHANDRAN; FOO; SUNDARARAJAN, 1996, p.6).

Logo, a etapa consiste em uma sequência de cálculos que visam atualizar os pesos sinápticos. Após esse ajuste realizado, o ciclo de *backpropagation* pode começar de novo ou não, dependendo da configuração dos parâmetros responsáveis por indicar o final para o ciclo.

Capítulo 4 - Realização do experimento

4.1. Ferramenta utilizada para implementação

Foi utilizado o pacote *scikit-learn*, na linguagem de programação Python, para a realização do experimento. Segundo Scikit-learn (c2022), esse é um projeto que começou a ser desenvolvido em 2007 e continua tendo atualizações periódicas até os dias atuais, com ajuda de sua comunidade internacional.

Esse módulo possui uma ampla gama de algoritmos de aprendizado de máquina para problemas supervisionados e não supervisionados. O foco do pacote é trazer a ferramenta de *Machine Learning* para não especialistas por meio do uso de uma linguagem de alto nível. É distribuído sob a licença BSD simplificada, visando encorajar o uso tanto no meio acadêmico quanto no meio comercial (PEDREGOSA, 2011, v.12, p.2825-2830).

O pacote disponibiliza tanto o algoritmo de Floresta Aleatória quanto o algoritmo de Redes Neurais Artificiais, nomeados, respectivamente, de "*RandomForestClassifier*" e "*MLPClassifier*".

4.2. Leitura dos dados

A coleção de dados, como descrita anteriormente, possui 2126 registros divididos em três classes de classificação.

Ademais, cada um desses registros possui 22 atributos, que são responsáveis por descrever as variáveis de características e a variável alvo.

Tabela 1 - Atributos do Conjunto de Dados	
	(continua)
Atributo	Descrição
baseline value (Valor da linha de base)	Representa a *Baseline Fetal Heart Rate* (FHR), ou em tradução livre, Frequência Cardíaca Fetal da Linha de Base. De modo geral, é a frequência cardíaca durante um segmento de 10 minutos arredondado para o incremento de 5 batidas por minuto mais próximo, com exclusão de alterações periódicas ou episódicas, períodos de variabilidade marcada e segmentos de linha de base que acabam por diferir em mais de 25 batidas por minuto (PERINATOLOGY.COM, c2014; ROBINSON, p.56-60).
accelerations (Acelerações)	Representa o número de acelerações por segundo. A aceleração é um crescimento repentino na taxa de FHR acima da linha de base com menos de 2 minutos de duração e com início ao pico de aceleração menor que 30 segundos (PERINATOLOGY.COM, c2014).
fetal_movement (Movimento fetal)	Representa o número de movimentos fetais por segundo
uterine_contractions (Contrações uterinas)	Representa o número de contrações uterinas por segundo
light_decelerations (Desacelerações leves)	Representa o número de desacelerações leves por segundo. As desacelerações são quantificadas pela profundidade do ponto mais baixo sob a linha de base, com duração quantificada em minutos e segundos desde o início ao fim da desaceleração, de maneira similar às acelerações (PERINATOLOGY.COM, c2014).
severe_decelerations (Desacelerações severas)	Representa o número de desacelerações severas por segundo.
prolongued_decelerations (Desacelerações prolongadas)	Representa o número de desacelerações prolongadas por segundo.
abnormal_short_term_variability (Variabilidade anormal de curto prazo)	Representa a percentagem de tempo com anormal variabilidade de curto prazo. A variabilidade é definida como flutuações na frequência cardíaca fetal de mais de 2 ciclos por minuto (PERINATOLOGY.COM, c2014).
mean_value_of_short_term_variability (Valor médio da variabilidade de curto prazo)	Representa o valor médio da variabilidade de curto prazo

Tabela 1 - Atributos do Conjunto de Dados	
(conclusão)	
Atributo	Descrição
percentage_of_time_with_abnormal_long_term_variability (Percentagem de tempo com variabilidade anormal de longo prazo)	Representa a percentagem de tempo com variabilidade anormal de longo prazo.
mean_value_of_long_term_variability (Valor médio da variabilidade de longo prazo)	Representa o valor médio da variabilidade de longo prazo.
histogram_width (Largura do histograma)	Representa a largura do histograma feito utilizando todos os valores de um registro.
histogram_min (Mínimo do histograma)	Representa o menor valor do histograma feito utilizando todos os valores de um registro.
histogram_max (Máximo do histograma)	Representa o maior valor do histograma feito utilizando todos os valores de um registro.
histogram_number_of_peaks (Número de picos do histograma)	Representa o número de picos do histograma feito utilizando todos os valores de um registro.
histogram_number_of_zeroes (Número de zeros do histograma)	Representa o número de zeros do histograma feito utilizando todos os valores de um registro.
histogram_mode (Moda do histograma)	Representa o valor da moda do histograma feito utilizando todos os valores de um registro.
histogram_mean (Média do histograma)	Representa o valor da média do histograma feito utilizando todos os valores de um registro.
histogram_median (Mediana do histograma)	Representa o valor da mediana do histograma feito utilizando todos os valores de um registro.
histogram_variance (Variância do histograma)	Representa o valor da variância do histograma feito utilizando todos os valores de um registro.
histogram_tendency (Tendência do histograma)	Representa o valor da tendência do histograma feito utilizando todos os valores de um registro.
fetal_health (Saúde fetal)	Representa a classe do registro para a classificação. É a variável alvo, com 3 classes de saúde fetal, descrita anteriormente.

Fonte (Adaptado): MARANHÃO, 2020

A Tabela 1 descreve os atributos utilizados no conjunto de dados, de maneira

resumida. As definições são simples e gerais, servindo para passar uma breve noção sobre a base de dados para proporcionar uma compreensão facilitada.

4.3. Padronizando os dados

4.3.1. Padronização e Normalização

A técnica de padronização e de normalização são técnicas de dimensionamento de características, pois a maioria dos algoritmos de aprendizado de máquina possui desempenho melhor se os dados de características estiverem na mesma escala (RASCHKA; MIRJALILI, 2017, p.120-121).

O algoritmo de *Random Forest* e o algoritmo de *Decision Tree* são exceções e não necessitam desse procedimento, pois eles são invariantes quanto a escala (RASCHKA; MIRJALILI, 2017, p.120).

A explicação da importância do procedimento se deve ao fato de que o algoritmo vai gastar mais tempo otimizando os pesos para erros de escalas grandes do que erros de escalas pequenas, devido à função de erro ao quadrado (RASCHKA; MIRJALILI, 2017, p.121).

A escolha pelo procedimento de padronização se deve a explicação de Raschka e Mirjalili (2017, p.121-122), a qual definiu que algoritmos de otimização, como o de descida do gradiente, funcionam melhor com esse procedimento do que com o método de normalização, devido ao fato de que esses algoritmos conseguem aprender os pesos com mais facilidade. O aprendizado é facilitado pela combinação de dois fatores: Os pesos sinápticos são iniciados, de modo geral, em valores próximos ou iguais a zero e a padronização centraliza os valores de características em média igual a 0 e desvio padrão igual a 1, fazendo com que as colunas de características tenham os mesmos parâmetros que uma distribuição normal. Enquanto isso, o algoritmo de normalização, normalmente, utiliza uma escala de 0 a 1.

4.3.2. Explicando a padronização

Figura 31 – Equação de Padronização

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

Fonte: RASCHKA; MIRJALILI, 2017, p.122

Na Figura 31 pode ser visualizada a equação utilizada para padronizar as informações de características de conjuntos de dados.

μ_x é a média da amostra de uma determinada coluna de características e σ_x é o seu correspondente desvio (RASCHKA; MIRJALILI, 2017, p.122).

A técnica foi aplicada na base de dados com o propósito de auxiliar o modelo de *Artificial Neural Network* a obter resultados melhores.

4.4. Divisão dos dados

Inicialmente, a base de dados foi dividida em duas coleções de dados: treinamento e teste. Com essa divisão, foram realizados testes iniciais com os algoritmos, apenas com o propósito de verificar um desempenho preliminar.

A técnica também é chamada de "*holdout cross validation*" (RASCHKA; MIRJALILI, 2017, p.190).

A partição do conjunto de dados foi de 25% dos registros para teste e 75% dos registros para treino.

Posteriormente, também foi utilizada a técnica de validação cruzada, que funciona de modo diferente e mais eficiente.

4.5. *Tuning* de parâmetros (Ajuste de parâmetros)

4.5.1. Explicação de *Grid Search* (Pesquisa via Grid)

Existem dois tipos de parâmetros para a técnica de *Machine Learning*: os que são aprendidos com a coleção de dados de treino e os parâmetros do algoritmo de aprendizagem. Esses últimos, por sua vez, são chamados de hiperparâmetros (RASCHKA; MIRJALILI, 2017, p.201).

Um exemplo de hiperparâmetro para o algoritmo de *Random Forest* é o critério de construção das árvores de decisão. Já para o algoritmo de *Artificial Neural Network*, o exemplo pode ser a função de ativação selecionada.

Assim, *Grid Search* consiste em um paradigma de busca exaustiva, em que é estabelecida uma lista de valores de diferentes hiperparâmetros e o computador testa o modelo com cada combinação desses hiperparâmetros para obter a combinação otimizada de valores (RASCHKA; MIRJALILI, 2017, p.201).

Assim, foi definida uma lista de hiperparâmetros para o algoritmo de *Random Forest* e para o algoritmo de *Artificial Neural Network* e a técnica se encarregou de buscar a melhor combinação dos hiperparâmetros passados na lista, possibilitando, assim, que os algoritmos obtenham melhores desempenhos.

4.5.2. Aplicação da técnica

Em cada um dos tópicos, será apresentada uma tabela com os hiperparâmetros fornecidos para a técnica de *Grid Search* na busca pelo melhor conjunto de hiperparâmetros em cada algoritmo. Uma breve explicação de cada parâmetro também será definida.

4.5.2.1. *Random Forest*

Tabela 2 – Conjunto de Parâmetros para *Grid Search* em *Random Forest*			
Parâmetro	Explicação do parâmetro	Possíveis valores	Hiperparâmetro selecionado
criterion (critério)	É a função responsável por medir a qualidade de uma divisão (SCIKIT-LEARN, c2022). Representa o critério de construção das árvores de decisão, explicado no Capítulo 2	*gini*, *entropy*	*entropy*
n_estimators (número de estimadores)	É o número de árvores na floresta (SCIKIT-LEARN, c2022). Representa o número de árvores de decisão que devem ser construídas para o algoritmo de *Random Forest* conseguir tomar sua decisão.	100, 300, 500, 700	100
min_samples_split (mínima divisão de amostras)	É o número mínimo de amostras requeridas para dividir um nódulo interno (SCIKIT-LEARN, c2022).	2, 4, 8	4
max_features (máximo de características)	É o número de características que devem ser consideradas quando ocorrer a busca pela melhor divisão (SCIKIT-LEARN, c2022). Quando se usa "*sqrt*", é utilizado a função de raiz quadrada no número de características para determinar a quantidade de atributos considerados e quando se usa "*log2*" é utilizado a função do logaritmo de 2 no número de características para a determinação (SCIKIT-LEARN, c2022).	*Sqrt, log2*	*sqrt*

Fonte: Elaboração Própria

4.5.2.2. *Artificial Neural Network*

Tabela 3 - Conjunto de Parâmetros para *Grid Search* em *Artificial Neural Network*

Parâmetro	Explicação do parâmetro	Possíveis valores	Hiperparâmetro selecionado
activation (ativação)	É a função de ativação para a camada oculta (SCIKIT-LEARN, 2022). As funções de ativação para as camadas ocultas foram explicadas no Capítulo 3.	*logistic, tanh, relu*	*tanh*
solver (solucionador)	É o responsável pela otimização dos pesos sinápticos (SCIKIT-LEARN, 2022). Os solucionadores "*sgd*" e "*adam*" são relacionados com técnicas de gradiente, enquanto o "*lbfgs*" vem de uma família de métodos nomeada "quasi-Newton" (SCIKIT-LEARN, c2022).	*sgd, adam, lbfgs*	*adam*
hidden_layer_sizes (tamanhos de camada oculta)	Representa o número de camadas ocultas que compõem a arquitetura de Rede Neural Artificial. Esse tópico também foi abordado no Capítulo 3.	(10,10), (50,50), (100,100)	(100,100)
tol (tolerância)	É a tolerância para a otimização. Quando a perda não está aumentando esse valor por n consecutivas interações, o treinamento para pois é considerado que se alcançou a convergência (SCIKIT-LEARN, c2022). O valor n de consecutivas interações é o próximo parâmetro da lista.	0.001, 0.0001, 0.00001	0.0001
n_iter_no_change (número de iterações sem mudança)	Representa o valor de n consecutivas interações citadas anteriormente. É o número máximo de épocas para não atender a melhoria, sendo eficaz apenas quando o solucionador for "*sgd*" ou "*adam*" (SCIKIT-LEARN, c2022).	10	10
max_iter (máximo de iterações)	É o número máximo de iterações. O solucionador itera até atingir a convergência, que é determinada pelo parâmetro "*tol*", ou até atingir o valor determinado de iterações (SCIKIT-LEARN, c2022).	500,1000, 1500	500

Fonte: Elaboração Própria

4.6. Validação cruzada

Foi utilizado o procedimento de “*K-fold cross validation*” para a realização da validação cruzada.

Nessa técnica, o conjunto de dados é dividido em k dobras sem substituições, sendo que k - 1 dobras serão utilizadas para treinamento do modelo e uma dobra é utilizada para testar o desempenho. O procedimento é repetido k vezes então é obtido k modelos e resultados (RASCHKA; MIRJALILI, 2017, p.191).

Assim, é obtida uma estimativa de performance menos sensível ao particionamento da base de treinamento do que o método de *holdout* (RASCHKA; MIRJALILI, 2017, p.192).

O número de dobras foi definido como 10 e o procedimento foi repetido por 30 vezes, visando obter uma média estável de todos os resultados obtidos.

4.7. Resultados obtidos

Foram realizados 3 tipos de testes durante a execução do experimento, visando averiguar os resultados obtidos pelos modelos.

O primeiro teste foi realizado após a etapa de divisão de dados, para obtenção de uma acurácia inicial.

O segundo teste foi realizado com a técnica de *Grid Search*, que buscou a melhor combinação de hiperparâmetros e forneceu o melhor conjunto encontrado junto com a acurácia obtida.

O último teste realizado foi com a técnica de Validação Cruzada, usando dos parâmetros encontrados com a técnica de *Grid Search*, buscando visualizar a acurácia, precisão, *recall* e *f1-score* obtidos.

Tabela 4 - Resultados Obtidos						
Modelo	Teste inicial	*Grid Search*	Validação cruzada			
			Accuracy	*Precision*	*Recall*	*F1-Score*
Random Forest	93.80%	94.60%	93.85%	91.79%	85.89%	88.31%
Artificial Neural Network	91.17%	93.98%	92.75%	87.92%	85.83%	86.51%

Fonte: Elaboração Própria

Na Tabela 4, podem ser vistos os resultados obtidos para cada algoritmo e para cada teste no experimento realizado.

No próximo capítulo, serão explicadas as medidas utilizadas para averiguação do desempenho dos algoritmos. Ademais, também serão realizadas comparações entre os resultados obtidos e comparações com os resultados obtidos por outros autores, para possibilitar a efetivação de uma conclusão a respeito do caso de estudo.

Capítulo 5 – Análise dos resultados

5.1. Medidas de desempenho dos algoritmos

Buscando a ampliação do escopo de avaliação do desempenho, serão utilizadas quatro medidas para medir o desempenho dos algoritmos, pois cada um desses parâmetros permite a avaliação do desempenho de uma perspectiva diferente.

Com as medidas de precisão, *recall* e *f1-score*, foi utilizado o parâmetro de *macro* para a média, que, de acordo com Scikit-learn (c2022), é uma medida que calcula as métricas para cada classe e encontra a média não balanceada para os registros, ou seja, não leva em conta a proporção de registros de cada classe em conta. Como as três classes de registros são importantes e distintas, julgou-se como mais apropriado a escolha.

5.1.1. *Accuracy* (Acurácia)

Acurácia é uma medida que visa determinar a utilidade de um conjunto de dados (BHATNAGAR;MAHESHWARI, 2016, p. 414).

O valor da acurácia é dado pelo número de instâncias corretamente classificadas, multiplicado por 100 e dividido pelo número total de instâncias consideradas, obtendo assim o resultado, como mostrado na Figura 32.

Figura 32 - Fórmula do Cálculo de Acurácia

$$\text{Accuracy} = \frac{\text{Correctly classified instances}}{\text{Total number of Instance}} \text{ x } 100$$

Fonte: BHATNAGAR; MAHESHWARI, 2016, p. 414

5.1.2. *Precision* (Precisão)

A precisão representa a proporção de instâncias que realmente pertencem a uma determinada classe, com relação ao número de instâncias que foram classificadas como pertencentes a essa classe (BHATNAGAR;MAHESHWARI, 2016, p. 414).

Figura 33 - Fórmula do Cálculo de Precisão

$$\text{Precision} = \frac{t_p}{t_p + f_p}$$

Fonte: BHATNAGAR; MAHESHWARI, 2016, p. 414

Na fórmula apresentada na Figura 33, t_p representa o número de registros verdadeiramente positivos, registros classificados como positivos que são positivos, enquanto f_p representa o número de falsos positivos, registros classificados como positivos que são negativos (BHATNAGAR;MAHESHWARI, 2016, p. 414).

5.1.3. *Recall*

Recall é definido como a proporção de registros classificados em uma classe com relação ao número de registros que realmente pertencem a essa classe, medindo, assim, a completude dos resultados (BHATNAGAR;MAHESHWARI, 2016, p. 414).

Também é utilizado o nome de "*Sensitivity*" para essa medida, ou em português, "Sensividade".

A sua fórmula calcula a divisão do número total de positivos classificados corretamente pela soma dessa quantidade de reais positivos com a quantia de falsos negativos, registros classificados como negativos que são positivos (BHATNAGAR;MAHESHWARI, 2016, p. 414).

Figura 34 - Fórmula do Cálculo de *Recall*

$$\text{Recall} = \frac{t_p}{t_p + f_n}$$

Fonte: BHATNAGAR; MAHESHWARI, 2016, p. 414

A Figura 34 apresenta a fórmula do cálculo de *Recall*. Como é possível notar, possui grande semelhança com a fórmula de cálculo de precisão, apenas trocando o uso de falsos positivos por falsos negativos.

5.1.4. *F1-Score*

F1-Score é um tipo de medida que combina o *recall* com a precisão (BHATNAGAR;MAHESHWARI, 2016, p. 414).

Também pode ser chamada de “*F-measure”*. Sua fórmula pode ser observada na Figura 35.

Figura 35 - Fórmula do Cálculo de *F1-Score*

$$F - \text{measure} = \frac{(2 * \text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

Fonte: BHATNAGAR; MAHESHWARI, 2016, p. 414

5.2. Comparação dos resultados obtidos pelos algoritmos

No Capítulo 4, foram apresentados os resultados obtidos com os algoritmos de *Random Forest* e *Artificial Neural Network*. Desse modo, é possível compará-los, visando obter conclusões.

Primeiramente, é possível observar os resultados advindos do teste inicial (apresentados na Tabela 4). Apesar de representarem apenas uma informação preliminar, é possível observar que os resultados são relativamente altos. Entretanto, é necessário buscar uma base mais sólida para inferir alguma conclusão, visto que representam apenas dados de acurácia advindos de um simples teste, sem uma sustentação sólida provinda de outras verificações.

Com o segundo teste realizado, o de ajuste de parâmetros, foi possível selecionar os melhores hiperparâmetros para cada algoritmo. Além do mais, é possível notar que a acurácia veio a aumentar, quando comparado aos testes iniciais. Dessa forma, é possível ver que a seleção de parâmetros possibilitou que os algoritmos obtivessem resultados melhores.

Até esse momento, é possível observar que os dois algoritmos obtiveram resultados relativamente próximos, com o algoritmo de *Random Forest* obtendo acurácias um pouco superiores. Como dito anteriormente, não é possível inferir uma grande conclusão apenas com esses testes iniciais, mas é capaz de se presumir que ambos estão em paridade.

Um fator que acaba por influenciar significativamente na justeza dos testes iniciais é a aleatoriedade presente na técnica de *Machine Learning*, responsável por interferir nos resultados dos algoritmos.

A aleatoriedade no aprendizado de máquina é um importante elemento, pois ajuda a eliminar vieses inerentes e ajuda na construção de modelos com capacidade de generalização. Acontecimentos aleatórios ocorrem em diversos momentos, como na divisão do conjunto de dados para treino e teste, em que os dados são selecionados de maneira aleatória. Também ocorrem com o algoritmo de *Random Forest*, na criação de subamostras dos dados de treinamento para a construção das árvores de decisão, e com o algoritmo de *Artificial Neural Network*, em que os pesos iniciais e os vieses dos neurônios são escolhidos de modo aleatório (NAIR, 20–).

Por isso, é necessário o processo de validação cruzada, para assegurar resultados mais sólidos e menos sensíveis.

Com os testes de validação cruzada executados, é possível inferir alguma informação. De modo geral, os algoritmos mantiveram as suas medidas de desempenho acima de 85%, sendo esse um indicativo que mostra que ambos os algoritmos têm um bom desempenho para o conjunto de dados considerado.

É possível observar que *recall* foi a medida com menor valor. Uma possível explicação é o fato de a base de dados ser desbalanceada, de modo que há muito mais registros de classe normal do que registros de classe patológica, tornando, assim, o algoritmo mais capacitado para classificar registros da classe normal, devido a estarem presente em maior quantidade na base de treinamento. Esse fator também vem a interferir nas outras medidas.

Contudo, apesar da diferença na proporção de registros, não é um fator que impossibilita os algoritmos de *Machine Learning* de conseguirem classificar registros da classe patológica. Além disso, apesar da queda do valor, quando comparado a outras medidas, ainda é um resultado considerável.

Por fim, foi possível notar que o algoritmo de *Random Forest* conseguiu um desempenho melhor que o algoritmo de *Artificial Neural Network* nos quatro tipos de medidas, o que parece ser um indicativo de sua superioridade. No entanto, para comprovar esse fato, seria necessário realizar um experimento comparativo mais aprofundado, por exemplo, considerando vários conjuntos de dados distintos originados de diferentes contextos de aplicação.

5.3. Comparação dos resultados obtidos por pesquisadores

Os estudos selecionados para a realização da comparação são os realizados por Bhatnagar e Maheshwari (2016) e Subha *et al.* (2013).

Ambos os estudos utilizaram o mesmo conjunto de dados que foi utilizado neste trabalho. Sendo assim, considerar também os resultados desses estudos é uma boa forma de corroborar os resultados obtidos com esse trabalho e embasar suas conclusões finais.

É necessário observar que cada estudo utiliza diferentes abordagens nos seus testes, como diferentes números de atributos e registros para a verificação de desempenho. Assim, será apenas realizada uma breve comparação em um alcance possível.

Bhatnagar e Maheshwari (2016) utilizaram as quatro medidas de desempenho usadas no teste realizado neste estudo. Contudo, ao invés de utilizar algum critério de média agrupada para precisão, *recall* e *f1-score*, os valores de classe foram separados.

No âmbito da acurácia, o algoritmo de *Random Forest* utilizado nesse trabalho ficou com valores próximos aos algoritmos de maior desempenho utilizado no estudo dos pesquisadores, superando, inclusive, o teste realizado com o modelo de *Random Forest* no estudo.

O algoritmo de *Artificial Neural Network* utilizado nesse trabalho, por sua vez, apesar de não ter ficado tão distante dos resultados obtidos por Bhatnagar e Maheshwari, não obteve resultados chamativos, sendo ultrapassado pela sua contraparte utilizada no estudo dos pesquisadores.

No quesito precisão, na pesquisa de Bhatgnar e Maheshwari, é possível observar que os algoritmos obtiveram altos desempenhos, principalmente para a classe normal e a classe patológica. Para o algoritmo de *Random Forest* do estudo dos pesquisadores, realizando uma média dos valores obtidos pelas classes, é possível constatar o valor de 92.50%, obtendo, assim, um desempenho maior que o algoritmo utilizado neste trabalho, que ficou com média de 91.79%.

Quando se procura os algoritmos com maior precisão do estudo dos pesquisadores, como o J48, com média de valores de classe de 93.17%, é possível constatar um desempenho de patamar ainda mais elevado. As Redes Neurais Artificiais utilizadas nesse trabalho também podem ser comparadas da mesma forma, mostrando que apesar do desempenho próximo, ainda não obtiveram resultados significativamente melhores.

Com a medida *recall* e *f1-score* podem ser realizadas comparações similares também para obtenção de resultados próximos.

Por outro lado, Subha *et al.* (2013), em seus estudos, utilizaram três das quatro medidas empregadas neste estudo, sendo elas: acurácia, *recall* e *f1-score*. Em seus estudos, os pesquisadores agruparam as classes em uma média só. Também realizaram testes com menos atributos, mostrando um aumento do desempenho inicial.

Comparando com os resultados obtidos, apesar de próximo, o desempenho no quesito acurácia do algoritmo de *Random Forest* deste estudo obteve resultados superiores àqueles apresentados no estudo pelos pesquisadores, superando até o algoritmo com mais desempenho, o modelo de Árvore de Decisão. Já o algoritmo de Redes Neurais Artificiais utilizado neste estudo obteve resultados próximos, e conseguiu superar a sua contraparte utilizada no estudo dos pesquisadores, a qual obteve valor de 92.60%.

Com relação às outras medidas, é possível notar que o algoritmo de Árvore de Decisão e o algoritmo de Redes Neurais Artificiais utilizado pelos pesquisadores obtiveram resultados mais regulares e superiores aos algoritmos utilizados nesse estudo.

A utilização do pacote “Weka”, em contraposição ao “*Scikit-learn*”, para implementação dos algoritmos pelos pesquisadores pode ser um dos fatores que permitiu a obtenção de resultados mais constantes, ou ainda, o processo de seleção de atributos também pode ter sido uma influência. Contudo, apenas uma abordagem comparativa mais rigorosa e aprofundada, com uma base de dados mais extensa e com diferentes cenários de abordagem poderia explicar de maneira mais precisa os resultados.

Figura 36 - Resultados de Bhatgnar e Maheshwari

Algorithm	MAE	Kappa Statistics	Accuracy
J48 Classifier	0.0408	0.8716	94.33 %
JRIP Classifier	0.0595	0.8339	92.74 %
Naive Bayes	0.1167	0.6221	82.31 %
Random Forest	0.0776	0.8423	93.20 %
Classification Via Regression	0.0652	0.8606	94.45%
MLP	0.063	0.82	93.22
Attribute Selected Classifier	0.05	0.85	94.22%

Algorithm	Precision		
	Suspicious	Normal	Pathologic
J48	0.814	0.981	1.000
JRIP	0.804	0.959	1.000
Naïve Bayes	0.563	0.951	0.667
Random Forest	0.813	0.962	1.000
Classification Via Regression	0.873	0.962	0.944
MLP	0.882	0.954	0.818
Attribute Selected Classifier	0.884	0.954	0.961

Algorithm	Recall		
	Suspicious	Normal	Pathologic
J48	0.933	0.962	0.800
JRIP	0.876	0.953	0.829
Naïve Bayes	0.798	0.864	0.514
Random Forest	0.876	0.968	0.743
Classification Via Regression	0.842	0.982	0.817
MLP	0.767	0.980	0.878
Attribute Selected Classifier	0.814	0.980	0.890

Algorithm	F-Measure		
	Suspicious	Normal	Pathologic
J48	0.869	0.971	0.889
JRIP	0.839	0.956	0.906
Naïve Bayes	0.660	0.906	0.581
Random Forest	0.843	0.965	0.852
Classification Via Regression	0.857	0.972	0.944
MLP	0.820	0.967	0.847
Attribute Selected Classifier	0.848	0.967	0.924

Fonte (Adaptado): BHATNAGAR; MAHESHWARI, 2016, p. 416-417

Figura 37 - Resultados de Subha *et al.*

Metrics	Classifiers			
	NB	DT	MLP	RBF
Accuracy	83.9	93.3	92.6	87.2
Sensitivity	83.9	93.3	92.6	87.2
Specificity	88.3	93.2	92.4	89.0
F-Score	85.2	93.3	92.5	87.8

Fonte (Adaptado): SUBHA *et al.*, 2013, p. 278-279

Na Figura 36 e na Figura 37 é possível observar os resultados obtidos pelos pesquisadores apresentados. Como ressaltado anteriormente, cada pesquisador usa abordagens e métodos próprios para suas pesquisas, então a comparação foi realizada dentro do limite possível.

Capítulo 6 - Conclusão

Como elucidado no Capítulo 5, os algoritmos utilizados obtiveram desempenho satisfatório, com o modelo de *Random Forest* sendo um pouco superior.

Além disso, foi possível mostrar que os algoritmos implementados conseguem acertar em quantidade considerável de casos, e como mostrado na comparação com outros estudos, seus resultados são compatíveis com pesquisas anteriores realizadas.

Assim, em uma situação real, apesar de não ser assegurado um desempenho de igual qualidade, devido à diferença de situação e questões levantadas anteriormente, como o *overfitting*, é possível supor que seria uma grande contribuição para a problemática.

Como mostrado por Bhatnagar e Maheshwari (2016), os algoritmos ainda conseguem um desempenho muito bom quando os registros são de classe patológica, o foco do problema, embora o melhor desempenho deles ocorra para a categoria normal.

Ademais, também foi constatado que diversos algoritmos conseguem obter bons resultados nessa base de dados, visto os valores de medidas com resultados superiores a 85%.

Referências Bibliográficas

AGHDAM, Hamed Habibi; HERAVI, Elnaz Jahani. Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification. Berlim: Springer International Publishing, 2017. Disponível em: https://www.google.com.br/books/edition/Guide_to_Convolutional_Neural_Networks/72UkDwAAQBAJ. Acesso em: 30 jun. 2022.

AGGARWAL, Charu C. Data Classification: Algorithms and Applications. Boca Raton: CRC Press, 2014. Disponível em: https://www.google.com.br/books/edition/Data_Classification/qm_SBQAAQBAJ. Acesso em: 28 jun. 2022.

AGGARWAL, Charu C. Neural Networks and Deep Learning: A Textbook. Berlim: Springer International Publishing, 2018. Disponível em: https://www.google.com.br/books/edition/Neural_Networks_and_Deep_Learning/achqDwAAQBAJ. Acesso em: 29 jun. 2022.

BERMAN, Jules J. Principles of Big Data: Preparing, Sharing, and Analyzing Complex Information. Waltham: Elsevier Science, 2013. Disponível em: https://www.google.com.br/books/edition/Principles_of_Big_Data/gEho0DI8a2kC. Acesso em: 29 jun. 2022.

BHATNAGAR, D.; MAHESHWARI, P. Classification of Cardiotocography Data with WEKA. International Journal of Computer Science and Network, v. 5, n. 2, p. 412–418, 2016. Disponível em: <eprints.rclis.org/29886/>. Acesso em: 14 abr. 2022.

BONACCORSO, Giuseppe. Machine Learning Algorithms. Birminghamh: Packt Publishing, 2017. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=_-ZDDwAAQBAJ&pg=PA9. Acesso em: 24 mar. 2022.

BRASIL. Ministério da Saúde. Brasil reduziu 8,4% a razão de mortalidade materna e investe em ações com foco na saúde da mulher. 2020. Disponível em: https://aps.saude.gov.br/noticia/8736. Acesso em: 22 mar. 2022.

BRASIL. Ministério da Saúde. Manual dos Comitês de Mortalidade Materna. 2007. Disponível em: https://bvsms.saude.gov.br/bvs/publicacoes/comites_mortalidade_materna_3ed.pdf. Acesso em: 22 mar. 2022.

BRASIL. Ministério da Saúde. Saúde Brasil 2018. 2018. Disponível em: https://bvsms.saude.gov.br/bvs/publicacoes/saude_brasil_2018_analise_situacao_saude_doencas_agravos_cronicos_desafios_perspectivas.pdf. Acesso em: 22 mar. 2022.

CAMPOS, D. Ayres-de *et al*. Sisporto 2.0: a program for automated analysis of cardiotocograms. Journal of Maternal-Fetal Medicine, v. 9, n. 5, p. 311–318, 2000.

FURLEY, Pedro Rogério. Cardiotocografia Prática - Anteparto e Intraparto. Rio de Janeiro: Editora Rubio, 2012. Disponível em: https://books.google.com.br/books?id=lUu2AwAAQBAJ&pg=PA28. Acesso em: 24 mar. 2022.

GLADWIN, C. H. Ethnographic Decision Tree Modeling. Thousand Oaks: SAGE Publications, 1989. Disponível em: https://www.google.com.br/books/edition/Ethnographic_Decision_Tree_Modeling/01pQc3h9yNMC. Acesso em: 29 jun. 2022.

HAYKIN, Simon. Redes Neurais: Princípios e Prática. 2. ed. Porto Alegre: Bookman Editora, 2007. Disponível em: https://www.google.com.br/books/edition/Redes_Neurais/bhMwDwAAQBAJ?hl=pt-BR&gbpv=0. Acesso em: 29 jun. 2022.

JADHAV, Pratibbha Vijay; PATIL, Vaishali Vilas. APPLICATION OF DECISION TREE FOR DEVELOPING ACCURATE PREDICTION MODELS. Raleigh: Lulu Publication, 2022. Disponível em: https://books.google.com.br/books?id=b3N2EAAAQBAJ. Acesso em: 28 jun. 2022.

KOZAK, Jan. Decision Tree and Ensemble Learning Based on Ant Colony Optimization. Berlim: Springer International Publishing, 2018. Disponível em: https://www.google.com.br/books/edition/Decision_Tree_and_Ensemble_Learning_Base/dSVhDwAAQBAJ. Acesso em: 28 jun. 2022.

MARANHÃO, Andrew. Fetal Health Classification. 2020. Disponível em: https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification. Acesso em: 18 jul. 2022.

MOOLAYIL, Jojo. Smarter Decisions – The Intersection of Internet of Things and Decision Science. Birmingham: Packt Publishing, 2016. Disponível em: https://www.google.com.br/books/edition/Smarter_Decisions_The_Intersection_of_In/1s2qDQAAQBAJ. Acesso em: 29 jun. 2022.

NAIR, Aashish. Harnessing Randomness in Machine Learning. 20--. Disponível em: https://towardsdatascience.com/harnessing-randomness-in-machine-learning-59e26e82fdfc. Acesso em: 5 set. 2022.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. Maternal Mortality. 2019. Disponível em: https://www.who.int/news-room/fact-sheets/detail/maternal-mortality. Acesso em: 22 mar. 2022.

PANESAR, Arjun. Machine Learning and AI for Healthcare: Big Data for Improved Health Outcomes. New York: Apress, 2019. Disponível em: https://www.google.com.br/books/edition/Machine_Learning_and_AI_for_Healthcare/ynuGDwAAQBAJ. Acesso em: 29 jun. 2022.

PEDREGOSA, Fabian *et al*. Scikit-learn: Machine Learning in Python. JMLR, v. 12, p. 2825-2830, 2011. Disponível em: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html. Acesso em: 4 jul. 2022.

PERINATOLOGY.COM. Intrapartum Fetal Heart Rate Monitoring. c2014. Disponível em: https://www.perinatology.com/Fetal%20Monitoring/Intrapartum%20Monitoring.htm. Acesso em: 18 jul. 2022.

RASCHKA, Sebastian; MIRJALILI, Vahid. Python Machine Learning. Birminghamh: Packt Publishing, 2017. Disponível em:

https://www.google.com.br/books/edition/Python_Machine_Learning/_plGDwAAQBAJ. Acesso em: 4 jul. 2022.

RAVICHANDIRAN, Sudharsan. Hands-On Deep Learning Algorithms with Python: Master Deep Learning Algorithms with Extensive Math by Implementing Them Using TensorFlow. Birmingham: Packt Publishing, 2019. Disponível em: https://www.google.com.br/books/edition/Hands_On_Deep_Learning_Algorithms_with_P/8DqlDwAAQBAJ. Acesso em: 1 jul. 2022.

REBALA, Gopinath; RAVI, Ajay; CHURIWALA, Sanjay. An Introduction to Machine Learning. Berlim: Springer International Publishing, 2019. Disponível em: https://www.google.com.br/books/edition/An_Introduction_to_Machine_Learning/u8OWDwAAQBAJ. Acesso em: 29 jun. 2022.

ROBINSON, Barrett. A Review of NICHD Standardized Nomenclature for Cardiotocography: The Importance of Speaking a Common Language When Describing Electronic Fetal Monitoring. Rev Obstet Gynecol, Chicago, p. 56-60, 2008. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2505172/. Acesso em: 18 jul. 2022.

SARATCHANDRAN, P; FOO, Shou King; SUNDARARAJAN, Narasimman. Parallel Implementations Of Backpropagation Neural Networks On Transputers: A Study Of Training Set Parallelism. Danvers: World Scientific Publishing Company, 1996. Disponível em: https://www.google.com.br/books/edition/Parallel_Implementations_Of_Backpropagat/3_jsCgAAQBAJ. Acesso em: 1 jul. 2022.

SCIKIT-LEARN. About us. c2022. Disponível em: https://scikit-learn.org/stable/about.html. Acesso em: 4 jul. 2022.

SCIKIT-LEARN. Sklearn.ensemble.RandomForestClassifier. c2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Acesso em: 23 jul. 2022.

SCIKIT-LEARN. Sklearn.metrics.recall_score. c2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html. Acesso em: 28 set. 2022.

SCIKIT-LEARN. Sklearn.neural_network.MLPClassifier. c2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. Acesso em: 23 jul. 2022.

SILVA, Ivan Nunes da *et al*. Artificial Neural Networks: A Practical Course. Berlim: Springer International Publishing, 2016. Disponível em: https://www.google.com.br/books/edition/Artificial_Neural_Networks/DL_mDAAAQBAJ. Acesso em: 29 jun. 2022.

SINGH, Manpreet; CHHABRA, Jitender Kumar. Software Fault Prediction Using Machine Learning Models and Comparative Analysis. *In*: FUTURISTIC Trends in Network and Communication Technologies. Singapura: Springer Nature Singapore, 2021. Disponível em:

https://www.google.com.br/books/edition/Futuristic_Trends_in_Network_and_Communi/d7AmEAAAQBAJ. Acesso em: 29 jun. 2022.

SUBHA, V. et al. Comparative Analysis of Classification Techniques using Cardiotocography Dataset. International Journal of Research in Information Technology, v. 1, n. 12, p. 274–280, 2013. Disponível em: https://www.researchgate.net/profile/Subha_v2/publication/322100603_Comparative_Analysis_of_Classification_Techniques_using_Cardiotocography_Dataset/links/5a44faf4458515f6b05460da/Comparative-Analysis-of-Classification-Techniques-using-Cardiotocography-Dataset.pdf. Acesso em: 14 abr. 2022.

UNICEF *et al*. A Neglected Tragedy. 2020. Disponível em: https://data.unicef.org/resources/a-neglected-tragedy-stillbirth-estimates-report/. Acesso em: 22 mar. 2022.

Apêndice – Código-fonte desenvolvido

```
#Lendo os dados

import pandas as pd

data = pd.read_csv('fetal_health.csv')

X_data = data.iloc[:,0:21]

X_data2 = data.iloc[:,0:21]

y_data = data.iloc[:,21:]


#Padronizando os dados

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_data = scaler.fit_transform(X_data)


#Dividindo em base de treino e base de teste

from sklearn.model_selection import train_test_split

X_data_train, X_data_test, y_data_train, y_data_test = train_test_split(X_data, y_d
ata, test_size = 0.25)


#Testes simples com os algoritmos

from sklearn.metrics import accuracy_score

from sklearn.ensemble import RandomForestClassifier

randomForest = RandomForestClassifier()

randomForest.fit(X_data_train, y_data_train.values.ravel())

previsoes = randomForest.predict(X_data_test)

print(accuracy_score(y_data_test, previsoes))

from sklearn.neural_network import MLPClassifier

rede_neural = MLPClassifier()

rede_neural.fit(X_data_train, y_data_train.values.ravel())

previsoes = rede_neural.predict(X_data_test)
```

```
print(accuracy_score(y_data_test, previsoes))


#Tuning de Parâmetros

parametros = {'criterion': ['gini', 'entropy'],

              'n_estimators': [100,300,500, 700],

              'min_samples_split': [2, 4, 8],

              'max_features': ['sqrt', 'log2']}

grid_search = GridSearchCV(estimator=RandomForestClassifier(), param_grid=parametro
s)

grid_search.fit(X_data_train, y_data_train.values.ravel())

print(grid_search.best_params_)

print(grid_search.best_score_)

parametros = {'activation':('logistic', 'tanh', 'relu'),

           'solver':('lbfgs', 'sgd', 'adam'),

            'hidden_layer_sizes':((10,10),(50,50),(100,100)),

            'tol': [0.001, 0.0001, 0.00001],

            'n_iter_no_change': [10],

            'max_iter': [500,1000,1500]}

grid_search = GridSearchCV(estimator=MLPClassifier(), param_grid=parametros)

grid_search.fit(X_data_train, y_data_train.values.ravel())

print(grid_search.best_params_)

print(grid_search.best_score_)


#Validação Cruzada

from sklearn.model_selection import  KFold

from sklearn.model_selection import cross_validate

from sklearn.metrics import make_scorer

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.metrics import precision_score
```

```
accuracyNeuralNetwork = []

precisionNeuralNetwork = []

f1NeuralNetwork = []

recallNeuralNetwork = []

accuracyRandomForest = []

precisionRandomForest = []

f1RandomForest = []

recallRandomForest = []

scoring = {'accuracy': 'accuracy',

           'precision': make_scorer(precision_score, average='macro'),

           'recall': make_scorer(recall_score, average='macro'),

           'f1': make_scorer(f1_score, average='macro')}

for i in range(30):

  kfold = KFold(n_splits=10, shuffle=True, random_state=i)

  randomForest = RandomForestClassifier(criterion= 'entropy', max_features= 'sqrt',
min_samples_split= 4, n_estimators= 100)

  scores = cross_validate(randomForest, X_data_train, y_data_train.values.ravel(),
cv = kfold, scoring=scoring)

  for value in scores['test_accuracy']:

    accuracyRandomForest.append(value)

  for value in scores['test_precision']:

    precisionRandomForest.append(value)

  for value in scores['test_recall']:

    recallRandomForest.append(value)

  for value in scores['test_f1']:

    f1RandomForest.append(value)

  neuralNetwork = MLPClassifier(activation= 'tanh', hidden_layer_sizes =(100, 100),
max_iter= 500, n_iter_no_change= 10, solver= 'adam', tol= 0.0001)

  scores = cross_validate(neuralNetwork, X_data_train, y_data_train.values.ravel(),
 cv = kfold, scoring=scoring)

  for value in scores['test_accuracy']:

    accuracyNeuralNetwork.append(value)
```

```
  for value in scores['test_precision']:
    precisionNeuralNetwork.append(value)
  for value in scores['test_recall']:
    recallNeuralNetwork.append(value)
  for value in scores['test_f1']:
    f1NeuralNetwork.append(value)
print(sum(accuracyRandomForest)/len(accuracyRandomForest))
print(sum(precisionRandomForest)/len(precisionRandomForest))
print(sum(recallRandomForest)/len(recallRandomForest))
print(sum(f1RandomForest)/len(f1RandomForest))
print(sum(accuracyNeuralNetwork)/len(accuracyNeuralNetwork))
print(sum(precisionNeuralNetwork)/len(precisionNeuralNetwork))
print(sum(recallNeuralNetwork)/len(recallNeuralNetwork))
print(sum(f1NeuralNetwork)/len(f1NeuralNetwork))
```