UNIVERSIDADE FEDERAL DE SANTA MARIA

COLÉGIO POLITÉCNICO

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

Alisson Billig Kroth

TÉCNICAS DE MACHINE LEARNING APLICADAS NO PROCESSO DE CERTIFICAÇÃO DE GRANJAS AVÍCOLAS

Santa Maria, RS

2023

Alisson Billig Kroth

TÉCNICAS DE MACHINE LEARNING APLICADAS NO PROCESSO DE CERTIFICAÇÃO DE GRANJAS AVÍCOLAS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Sistemas para Internet da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Tecnólogo em Sistemas para Internet**.

Orientador: Prof. Dr. Alencar Machado

Santa Maria, RS

2023

Alisson Billig Kroth

TÉCNICAS DE MACHINE LEARNING APLICADAS NO PROCESSO DE CERTIFICAÇÃO DE GRANJAS AVÍCOLAS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Sistemas para Internet da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Tecnólogo em Sistemas para Internet.**

Aprovado em 4 de julho de 2023.

_____

Alencar Machado, Dr. (Presidente/Orientador)

_____

Vinícius Maran, Dr. (UFSM)

_____

Glênio Descovi de Freitas, Me.

Santa Maria, RS

2023

RESUMO

TÉCNICAS DE MACHINE LEARNING APLICADAS NO PROCESSO DE CERTIFICAÇÃO DE GRANJAS AVÍCOLAS

AUTOR: Alisson Billig Kroth

ORIENTADOR: Prof.º Dr. Alencar Machado

Nos últimos anos a área de Inteligência Artificial (IA) tem demonstrado muitos avanços, principalmente com o uso de machine learning e redes neurais, gerando grandes oportunidades de pesquisa. Boa parte da ampliação da área de IA deve-se ao aumento na utilização das técnicas de machine learning, as quais têm como objetivo treinar computadores para que sejam capazes de fazer previsões e tomar decisões a partir de um conjunto de dados. Este estudo tem como propósito encontrar padrões e aplicar técnicas de *machine learning* utilizando uma base de dados da Plataforma de Defesa Sanitária Animal do Rio Grande do Sul (PDSA-RS) com a finalidade de automatizar operações e/ou auxiliar usuários a tomarem decisões mais precisas e inteligentes no processo de certificação de aves do sistema. O tipo de aprendizado utilizado é o supervisionado com foco em classificação, onde o algoritmo é treinado através de um conjunto de dados e tenta prever qual é a classe de uma nova instância. Já os algoritmos estudados e utilizados são (a) *Support Vector Machine* (SVM) o qual é excelente para classificação binária e linear, mas também com a possibilidade de classificação multiclasse com o uso do kernel trick. (b) *K-nearest neighbors* (KNN) que tem o funcionamento simples e uma ótima performance para a classificação multiclasse. (c) *Decision Tree*, sendo simples de entender como o modelo gerado funciona, porque, basicamente ele cria uma árvore de decisões baseada em regras que são aprendidas pelo algoritmo na etapa de treino, facilitando o entendimento humano sob o funcionamento do modelo e (d) Redes Neurais que são um tipo de algoritmo mais robusto, com uma capacidade de aprendizado muito grande. A aplicação destas técnicas foi feita utilizando a linguagem de programação *Python* em conjunto com as bibliotecas Scikit-learn e TensorFlow em um conjunto de dados no formato CSV, extraídos de um banco de dados da PDSA-RS. A acurácia dos algoritmos tradicionais chegou no máximo de 79.88% com o KNN, já com redes neurais obteve-se o resultado de 94%.

Palavras-chave: *Decision Tree*. Inteligência Artificial. *K-nearest neighbors*. Redes Neurais. *Support Vector Machine*.

ABSTRACT

CERTIFICATION OF POULTRY FARM USING MACHINE LEARNING

AUTHOR: Alisson Billig Kroth

ADVISOR: Prof.º Dr. Alencar Machado

In recent years, the field of Artificial Intelligence (AI) has shown many advances, especially with the use of machine learning and neural networks, generating great research opportunities. A large part of the expansion of the AI area is due to the increase in the use of machine learning techniques, which aim to train computers to be able to make predictions and make decisions based on a set of data. This study aims to find patterns and apply machine learning techniques using a database from the Animal Health Defense Platform of Rio Grande do Sul (PDSA-RS) with the purpose of automating operations and/or assisting users to make more accurate and intelligent decisions in the bird certification process. The type of learning used is the supervised with a focus on classification, where the algorithm is trained through a set of data and tries to predict which class a new instance belongs to. The algorithms to be studied and used are (a) Support Vector Machine (SVM) which is excellent for binary and linear classification, but also with the possibility of multi-class classification with the use of the kernel trick. (b) K-nearest neighbors (KNN) which has a simple operation and a great performance for multi-class classification. (c) Decision Tree, being simple to understand how the generated model works, because, basically, it creates a decision tree based on rules that are learned by the algorithm in the training stage, facilitating human understanding of the model's operation and (d) Neural Networks which are a more robust type of algorithm, with a very large learning capacity. The application of these techniques was made using the Python programming language in addition with the libraries Scikit-learn and TensorFlow on a set of data in CSV format, extracted from a PDSA-RS database. The accuracy of the traditional algorithms reached a maximum of 79.88% with KNN, while with neural networks the result was 94%.

Keywords: Artificial Intelligence. Decision Tree. K-nearest neighbors. Neural Networks. Support Vector Machine.

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de sistemas para a internet tem mostrado muitos avanços, assim como a área de Inteligência Artificial (IA) que existe há bastante tempo, mas tem crescido muito nos últimos anos, principalmente com o uso de *machine learning* e redes neurais, gerando grandes oportunidades de pesquisa (Ludermir, 2021).

A área de *machine learning* é uma subarea de IA e tem como foco algoritmos capazes de serem treinados para aprender como executar tarefas utilizando um conjunto de dados ao invés de regras escritas “manualmente” em código que em vários casos podem se tornar algo muito complexo e dificil de ser feito e mantido por desenvolvedores (Payette & Lu, 2018).

Técnicas de *machine learning* tornam possível que uma máquina consiga aprender padrões e prever eventos a partir de um conjunto de dados, isto é, aprendem com a utilização de um grande volume de dados, por isso o crescimento no desenvolvimento de sistemas, os quais por sua vez geram dados, contribuem e estão fortemente relacionados com o avanço da IA (Ludermir, 2021). O conceito de IA já existe desde 1950, porém somente nos últimos anos algoritmos de IA e *machine learning* passaram a serem mais reconhecidos e bem sucedidos, isso deve-se principalmente ao aumento do poder computacional e também, a grande quantidade de dados que existem graças ao cresimento da utilização de sistemas em diversas areas (Ludermir, 2021).

Com estes avanços, os sistemas estão cada vez mais utilizando *machine learning* tanto para a descoberta de informação em busca de evoluir o modelo de negócio quanto para melhorar a experiência do usuário na utilização do sistema. A Plataforma de Defesa Sanitária Animal do Rio Grande do Sul (PDSA-RS) é um sistema que existe há alguns anos e possui uma base de dados relativamente grande com uma forte relação à área de medicina veterinária, sendo assim a aplicação de técnicas de *machine learning* neste sistema podem ser exploradas de forma promissora.

O sistema da PDSA-RS tem como objetivo sistematizar a cadeia de operações feitas dentre as empresas, laboratórios veterinários, Ministério da Agricultura, Pecuária e Abastecimento (MAPA) e o Serviço Veterinário Estadual para manter o controle da sanidade animal e rastrear a disseminação de doenças nas granjas agrícolas. A PDSA-RS possui um módulo que sistematiza o processo de certificação

de granjas de reprodução avícola, estas granjas passam por alguns procedimentos com a finalidade de serem certificadas livres de doenças por um órgão oficial. Um destes procedimentos envolve fazer a coleta de materiais que são enviados para um laboratório com a finalidade de determinar se algum núcleo da granja avícola está com uma contaminação de doenças, os materiais são coletados no contexto específico de cada núcleo, sendo assim, a aplicação de um algoritmo de *machine learning* pode ser muito útil para aprender os padrões e melhorar o processo na tomada de decisão a fim de tornar mais efetivo a inspeção nas granjas avícolas para aumentar o controle sobre a disseminação de doenças.

1.1 Objetivos

O trabalho tem como objetivo principal aplicar técnicas de *machine learning* com a utilização da linguagem *Python* e da biblioteca Scikit-learn em busca de encontrar um algoritmo de classificação que seja capaz de aprender a partir de um *dataset* extraído da base de dados do PDSA-RS, com o propósito de auxiliar usuários do sistema na tomada de decisão na etapa em que é feito a coleta dos materiais, observando diversos atributos referentes ao contexto em que o núcleo de aves se encontra e treinar um modelo que possa, a partir desse contexto classificar qual o grupo de materiais devem ser coletados. Para tanto, o trabalho possuí os seguintes objetivos específicos:

- Realizar um estudo teórico sobre *machine learning* e classificação;
- Estudar os algoritmos de classificação *Support Vector Machine* (SVM), *K-nearest neighbors* (KNN), *Decision Tree* e Redes Neurais;
- Fazer a extração dos dados na base de dados do PDSA-RS;
- Fazer o pré-processamento dos dados utilizando a linguagem *Python* e as bibliotecas *pandas* e *numpy*;
- Aplicar técnicas de classificação com os algoritmos estudados nos dados extraídos, utilizando as bibliotecas Scikit-learn para os algoritmos tradicionais e *TensorFlow* para Redes Neurais;
- Encontrar os hiperparâmetros ideais para cada algoritmo utilizado;
- Apresentar os resultados obtidos na utilização dos algoritmos.

2 REFERENCIAL TEÓRICO

2.1 Machine learning

Machine learning (aprendizado de máquina) é o aspecto primário que promove a condição de qualquer sistema ser considerado inteligente (Priya et al., 2020). O conceito da área de *machine learning* se resume em treinar computadores para que sejam capazes de fazerem previsões e tomar decisões a partir de um conjunto de dados (Shamrat et al., 2021). O principal objetivo de *machine learning* é estudar e melhorar modelos matemáticos que podem ser treinados com dados relacionados ao contexto, para tomar decisões sem ter um conhecimento completo de todos os fatores externos (Bonaccorso, 2017).

Existem diversos tipos de algoritmos para diferentes tipos de aprendizado como por exemplo: *Random Forest* (RF), *Naive Bayes*, *Support Vector Machine* (SVM), *Perceptron* (Redes Neurais) e *Decision Tree* que são alguns algoritmos de aprendizado supervisionado (Osisanwo et al., 2017). Não existe um melhor algoritmo ou um que se encaixa em todos os casos para resolver um problema. O algoritmo a ser utilizado depende do tipo de problema que se deseja resolver, o número de variáveis dentre outras questões (Mahesh, 2020).

2.1.1 Tipos de aprendizado

Existem três principais tipos de aprendizado de máquina: (a) aprendizado supervisionado, (b) aprendizado não-supervisionado e (c) aprendizado por reforço. O aprendizado supervisionado (Figura 1) é a metodologia de maior significância em *machine learning*, seu funcionamento implica em aprender um padrão aplicado em um conjunto de variáveis de entrada e de saída para prever as saídas não existentes no conjunto de dados (Cunningham et al., 2008).

No aprendizado não-supervisionado não há nenhuma saída específica de dados para o conjunto de treinamento, o objetivo do algoritmo de *machine learning* é extrair informações úteis usando apenas as variáveis de entrada (Glielmo et al., 2021). Aprendizado por reforço é sobre aprender o que fazer, como transformar situações em ações, assim como aumentar o sinal de recompensa numérica. O aprendiz não é informado quais ações realizar, mas invés disso ele deve descobrir quais ações rendem o melhor resultado ao testá-las (Mahesh, 2020; Sutton & Barto, 2018).

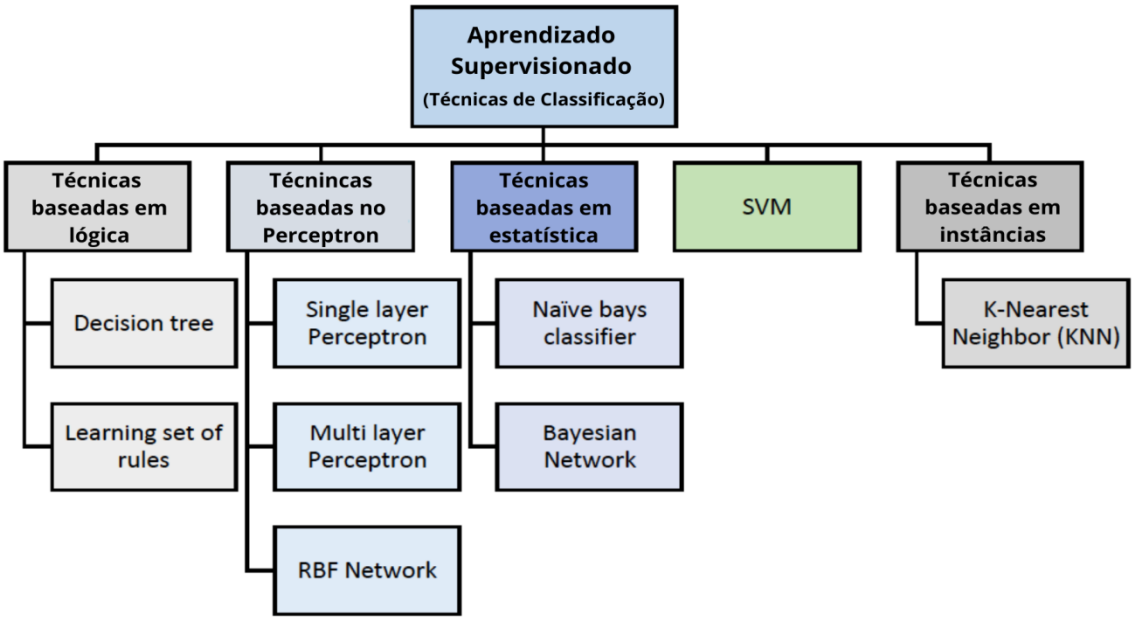

Figura 1 - Algoritmos de classificação de Aprendizado Supervisionado.
Fonte: Adaptado de: Soofi & Awan (2017).

2.1.2 Termos Utilizados

A área de *machine learning* possui muitos termos com importância no entendimento de como os algoritmos e as técnicas funcionam, tornando-se assim necessário a explicação dos termos utilizados durante o trabalho com uma lista que foi feita utilizando como referência o glossário de *machine learning* desenvolvido pelo Google Developers, 2022:

- *Feature*:
    - Uma variável de entrada mensurável utilizada para o aprendizado de um modelo de *machine learning*.
- *Label*:
    - No aprendizado supervisionado, considerado como um resultado ou a resposta de uma linha do *dataset*.
- *Dataset*:
    - Conjunto de dados normalmente no formato CSV. Também pode ser considerado como um conjunto que contém *features* e uma *label*.
- Classe:

    - No contexto de classificação uma classe é uma das categorias que uma *label* pertence.

- Classificação Binária:
    - Quando a *label* possui apenas 2 categorias diferentes.
- Classificação Multiclasse:
    - Quando a *label* possui mais de 2 categorias.
- *Feature space*:
    - Relativo ao número de dimensões em que as *features* se encontram sem incluir a *label*.
- *Overfitting*:
    - O modelo se ajusta demais com os dados de treino, prejudicando os resultados com dados de teste.
- *Underfitting*:
    - O modelo fica com uma acurácia abaixo do esperado. Por conta da falta de treino.
- Hiperparâmetro:
    - São parâmetros que impactam diretamente no funcionamento dos algoritmos, mudando a forma com que eles atuam e consequentemente os resultados.
- *Tuning*:
    - Processo de melhoria na utilização dos hiperparâmetros de um algoritmo.
- *TensorFlow*:
    - Uma plataforma de *machine learning* distribuída em grande escala e muito utilizada para algoritmos de Redes Neurais desenvolvida pela empresa Google.

2.2 Algoritmos de Classificação

A técnica de *machine learning* mais utilizada é a classificação, onde o algoritmo é treinado através de um conjunto de dados e tenta prever qual é a classe de uma nova instância, sendo assim uma ótima técnica para descoberta de conhecimento e previsões (Soofi & Awan, 2017).

Quando se aplica *machine learning* em um banco de dados real, o *dataset* pode aparecer de vários tipos e dimensões diferentes, sempre existem diversos problemas relacionados aos dados, como quando o *dataset* possui dados indesejados, nulos ou outliers (valores atípicos) que pode ser tanto um valor válido (porém fora da curva) quanto um ruído (valor incorreto). Além disso, normalmente os *datasets* possuem diversas *features* que tem pouca ou nenhuma relevância sendo também ruidosos, atrapalhando a criação de um modelo (Akpan & Starkey, 2021).

2.2.1 Support Vector Machine

O algoritmo de *Support Vector Machine* (SVM) é um algoritmo de aprendizado supervisionado muito utilizado, principalmente para classificação linear (Lorena & Carvalho, 2007). Foi desenvolvido a partir da teoria do aprendizado estatístico que tem como objetivo desenvolver ferramentas matemáticas e algoritmos para entender e analisar o aprendizado em um contexto estatístico, possuindo fundamentos teóricos utilizados em diversas técnicas de *machine learning* (Shalev-Shwartz & Ben-David, 2014).

Considerando uma classificação binária, o objetivo do algoritmo é traçar um hiperplano com a finalidade de separar os dados em duas classes, permitindo que com a inserção de um novo registro, possa ser previsto a qual grupo ele pertence, além disso, são criadas duas margens que definem o espaço entre as categorias, os pontos de dados (data points) que ficam sobre a margem são denominados suportes de vetor (Figura 2), quando o espaço dessa margem fica muito pequeno, normalmente pode-se considerar um *overfitting*, já quando as margens tem um espaço maior, o modelo terá melhor eficácia para fazer as previsões com dados de teste, porém normalmente ocorrem mais data points mal classificados. Normalmente é preciso fazer um balanço entre uma margem larga o suficiente, porém que não possua muitos data points mal classificados (IBM).

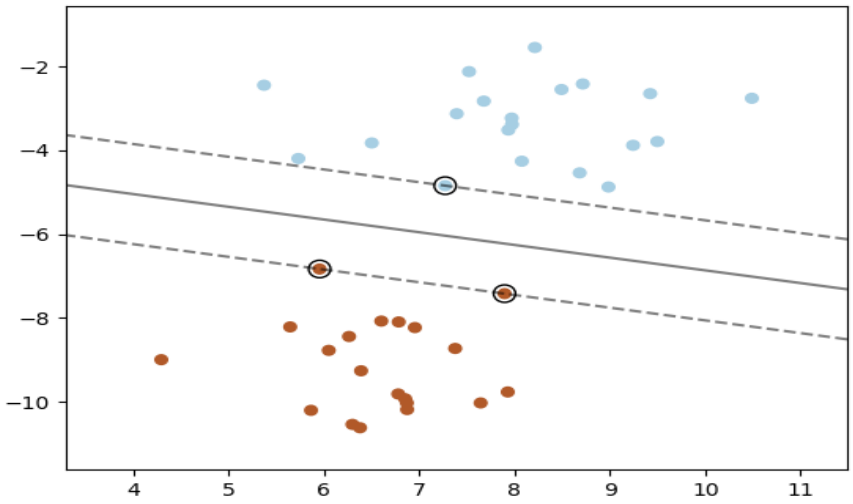

Figura 2 - Exemplo do funcionamento do SVM.
Fonte: *Scikit-learn*.

Em vários casos a classificação binária não é suficiente, tornando-se necessário uma classificação multiclasse, que aumenta a complexidade do problema e dificulta a utilização do SVM de forma linear. Além disso, também existem *datasets* com duas classes, mas que são impossíveis de serem divididos por uma reta, (Figura 3A), onde as classes ficam divididas de forma circular, impedindo que uma reta de separação seja efetiva, porém quando os dados estão separados em 3 dimensões (Figura 3B) é possível que uma reta possa fazer a separação das duas classes (Kim, 2015).

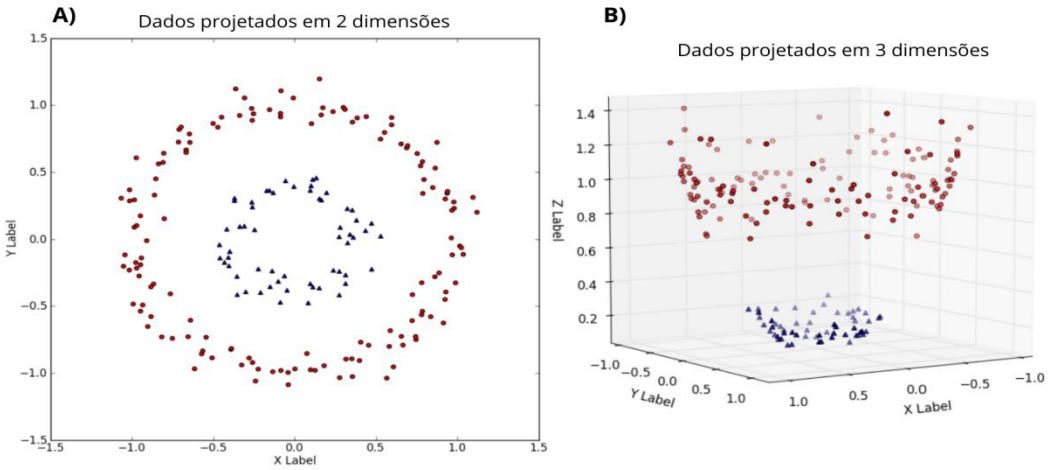

Figura 3 - *Dataset* não separável de forma linear. A) Dados projetados em 2 dimensões. B) Dados projetados em 3 dimensões.
Fonte: Adaptado de: Kim (2015).

Para resolver esse problema é possível aplicar a classificação não linear com o SVM, utilizando o chamado kernel trick (truque do kernel) que mapeia as entradas em *feature spaces* de alta dimensão (Mahesh, 2020). Basicamente o kernel trick converte os dados para um espaço dimensional maior e passa pelo processo de aprendizado neste espaço, faz uma reta linear separando as classes (Figura 4A) e após isso faz a projeção de volta para o espaço com duas dimensões, gerando por fim, uma classificação não linear (Figura 4B) (Kim, 2015).

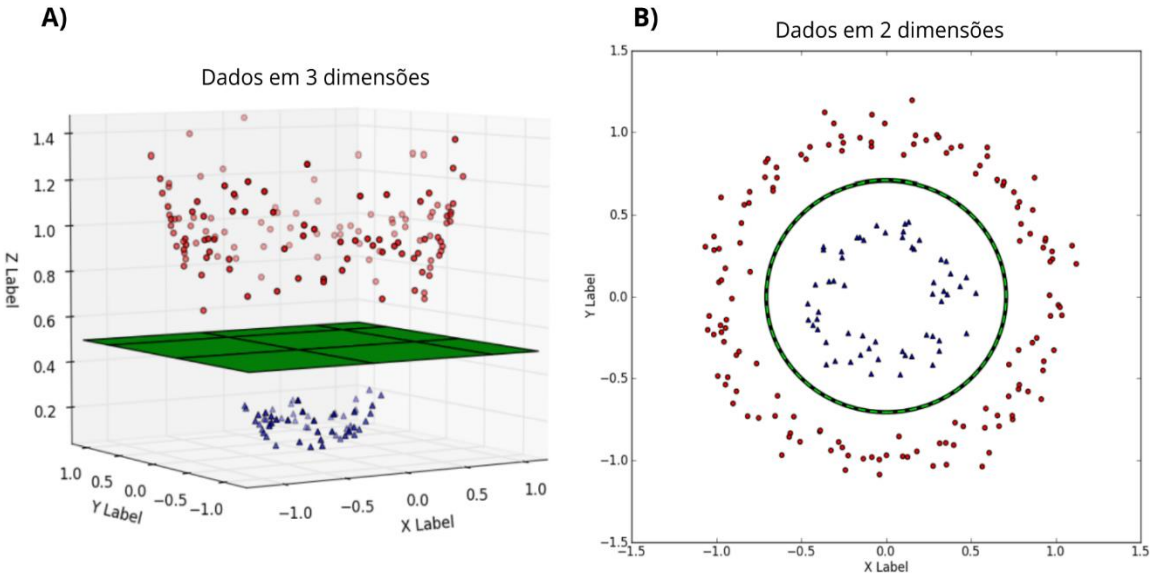

Figura 4 - Classificação utilizando o kernel trick. A) Classificação dos dados separados em 3 dimensões. B) Visualização dos dados em 2 dimensões após a classificação feita.
Fonte: Kim (2015).

2.2.2 K-Nearest Neighbors

O *K-Nearest Neighbors* (KNN) é um algoritmo que tem o funcionamento relativamente simples, é muito usado para classificação multiclasse, mas também pode ser usado para regressão. Normalmente os algoritmos de *machine learning* realizam ações na etapa de treino, porém o KNN tem o funcionamento um pouco diferente, pois ele apenas utiliza os dados de treino para classificar um novo elemento, porém na etapa de treino ele apenas salva em memória o conjunto de dados, ou seja, a predição é feita na etapa de teste.

Seu grande diferencial é que ele pode ter uma velocidade de execução muito rápida por conta de sua simplicidade, porém isso escala de forma linear de acordo com o número de colunas no *dataset*, ou seja, caso a quantidade de dados seja imensa, o algoritmo pode ser muito lento. A maneira que o KNN faz a classificação de

um novo dado é simples, basicamente ele calcula a distância entre os data points existentes utilizando o valor do hiperparâmetro K, e o classifica ao grupo com maior número de data points próximos (Taunk et al., 2019).

Como é possível visualizar na Figura 5, existem duas classes e um novo data point é inserido, então são feitos cálculos de distância com base em todos os dados que foram salvos na memória na etapa de treino para definir à qual classe ele pertence, depois levando em consideração o valor de K, o data point é atribuído a classe onde existe o maior número de data points próximos.

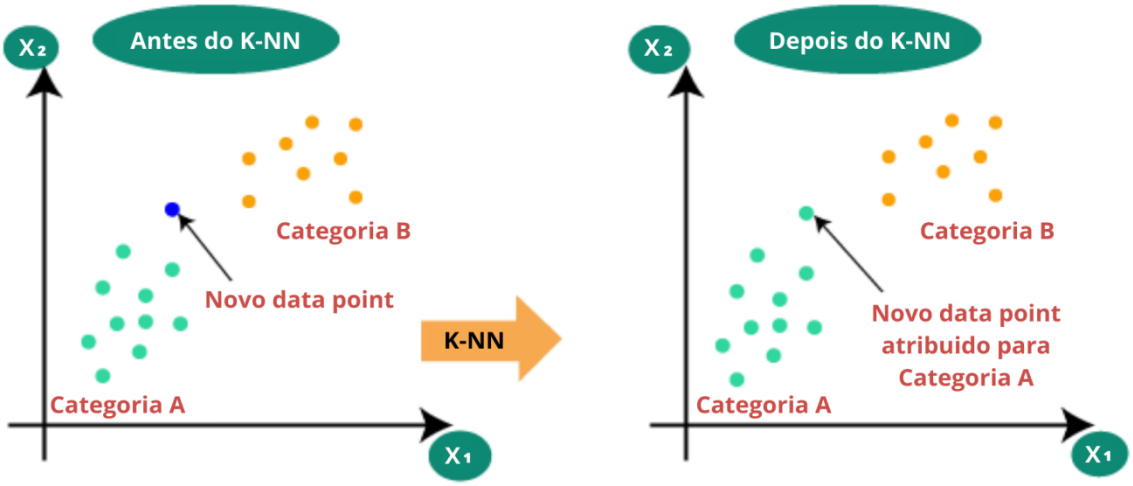

Figura 5 - Classificação básica com KNN.
Fonte: Adaptado de javatpoint.

Existem várias formas de calcular a distância entre os data points, porém a mais usada de todas é a distância euclidiana que calcula a medida da distância em linha reta entre dois pontos (Jain, 2022).

A escolha do valor de K tem um papel muito importante no funcionamento do algoritmo, pois ele implica em quantos pontos próximos devem ser levados em consideração para realizar a classificação. Um valor de k muito baixo pode causar *overfitting*, já o valor muito alto pode causar *underfitting*, ou seja, encontrar o valor de K não é uma tarefa muito simples (Jain, 2022).

Na Figura 6 é possível observar como o hiperparâmetro K influência na classificação de um novo data point, se o valor de K for 1 ele irá classificar como B pois leva em consideração apenas a distância de 1 data point e caso K seja 3 será classificado como A, pois levando em consideração a distância de 3 pontos, o número de pontos próximos da classe A é maior.

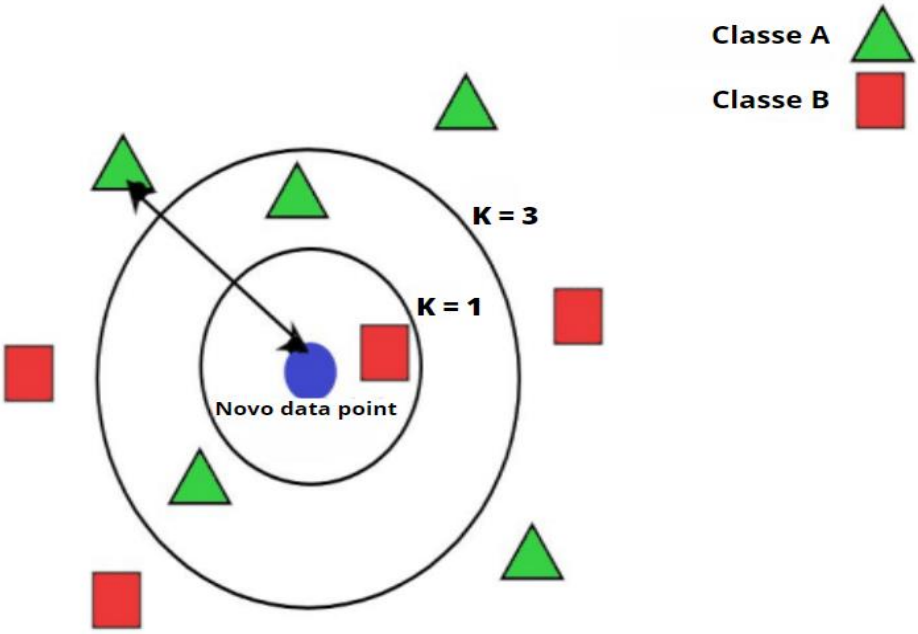

Figura 6 - Impacto do valor de K no KNN.
Fonte: Adaptado de Taunk et al., 2019.

O *elbow method* (método cotovelo) é uma forma muito utilizada para se encontrar um bom valor de K, o método calcula a taxa de erros utilizando vários valores de K diferentes, onde pode-se encontrar uma queda na taxa de erros, o ponto onde essa queda ocorre é chamado de *elbow point* (Figura 7), os valores de K próximos de onde ocorre a queda, normalmente são os melhores para serem utilizados (Jain, 2022).

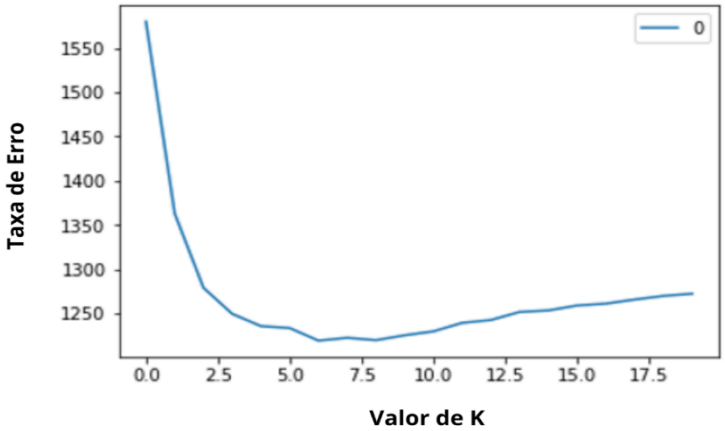

Figura 7 - Exemplo do e*lbow method*.
Fonte: Adaptado de Jain, 2022.

2.2.3 Decision Tree

Decision Tree é um algoritmo muito usado para a tomada de decisões em diversas áreas, na área de medicina por exemplo é muito usado por ter uma alta

acurácia em tarefas de classificação e uma representação do conhecimento adquirido de fácil entendimento (Podgorelec et al., 2002).

Assim como o KNN, *Decision Tree* pode ser usado tanto para classificação quanto para regressão, seu funcionamento de maneira geral se resume em criar uma árvore de decisões baseada em regras que são aprendidas pelo algoritmo na etapa de treino, essas regras ficam em um nó da árvore e têm um retorno booleano que é usado para comparar o data point a ser classificado na etapa de teste. O algoritmo começa em um nó raiz e passa para um nó interno e depois de um nó para outro fazendo comparações e decidindo qual caminho seguir até chegar em um nó que não possua nenhum nó filho (Figura 8), este nó é chamado de nó folha e é onde o algoritmo chega no resultado final da classificação (Jijo & Abdulazeez, 2021).

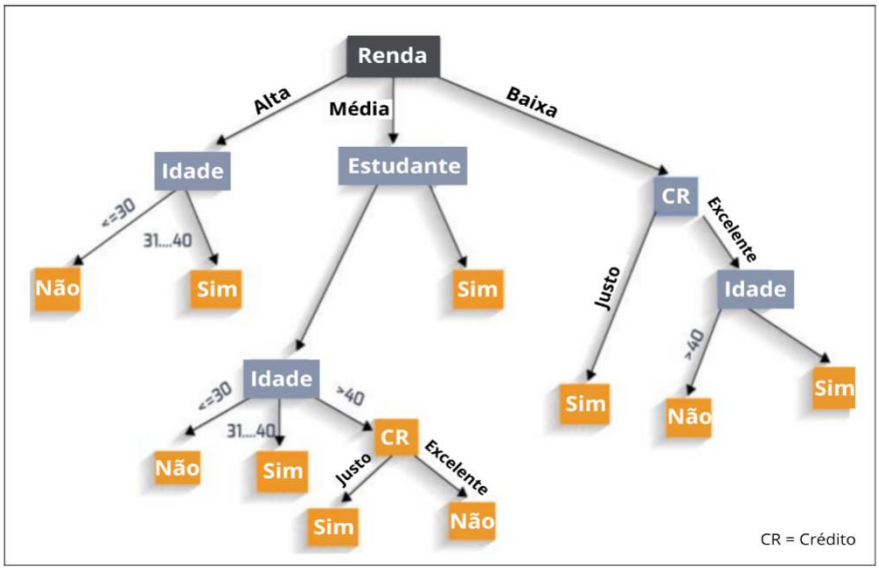

Figura 8 - Classificação com o algoritmo *Decision Tree.*
Fonte: Adaptado de Jijo & Abdulazeez, 2021.

Entropia, Ganho de Informação e Índice de Gini são medidas que variam entre 0 e 1 e são importantes para o algoritmo tomar decisões e separar corretamente os nós na etapa de treino. Entropia é usada para calcular a impureza do *dataset*, ou melhor, o grau de aleatoriedade dos dados relacionado à quantidade de *labels* que cada classe o *dataset* contém, ou seja, quanto menor a impureza, melhor para o algoritmo. O Ganho de Informação usa a entropia em sua fórmula e sua função é representar a diferença da entropia antes e depois de passar por um nó de decisão, ou seja, quanto maior o ganho de informação melhor para a classificação (Jijo & Abdulazeez, 2021). O Índice de Gini, também conhecido por Impureza de Gini, calcula

a probabilidade de uma *feature* aleatoriamente separada ser classificada incorretamente (Singh & Gupta, 2014).

Existem vários algoritmos de *Decision Tree*, cada tendo suas particularidades no funcionamento da forma com que o modelo encontra e gera as condições em cada nó de decisão, dentre os mais conhecidos e utilizados estão CART, ID3 e C4.5.

CART é um algoritmo utilizado tanto para regressão quanto para classificação, tem a forte característica de que gera árvores binárias pelo fato de que cada nó tem apenas dois nós filhos. Ele utiliza a medida do Índice de Gini para separar o *dataset*, procurando a melhor uniformidade nos nós filhos e gerar a árvore binária, uma vantagem que o CART tem em relação ao ID3 e o C4.5 é que ele consegue lidar melhor com a falta de dados.

O algoritmo ID3 é considerado simples, essencialmente ele gera a árvore testando os valores das propriedades, em cada nó da árvore uma propriedade é testada baseado em minimizar a Entropia e maximizar o Ganho de Informação e os resultados são utilizados para separar o *dataset*.

Com o funcionamento similar ao ID3, C4.5 é uma melhoria do algoritmo ID3, ele faz mais tentativas de encontrar uma propriedade com o Ganho de Informação maior, aumentando em alguns casos a assertividade, porém levando mais tempo para gerar a árvore e também gera uma árvore maior e mais complexa (Singh; Gupta, 2014).

2.2.4 Redes Neurais

As redes neurais são um tipo de algoritmo de *Machine Learning* inspirado pelo funcionamento e a estrutura do cérebro humano que consegue aprender a partir de dados. De maneira simplificada as redes consistem em neurônios interconectados que processam e transmitem informação por conexões que possuem peso e bias (desvio), passam pelas camadas e por funções de ativação até chegar em um resultado na camada de saída.

Nos últimos anos as redes neurais tem se provado ferramentas poderosas para várias tarefas em diversas áreas do conhecimento, principalmente com reconhecimento de imagens, de fala e processamento de linguagem natural. Seu grande diferencial de outros algoritmos é que possuem a habilidade de aprender e se adaptar para novas informações, diferente dos algoritmos tradicionais.

(Lecun et al., 2015).

As redes neurais funcionam através de camadas que contém neurônios, cada neurônio de uma camada se conecta com todos os neurônios da próxima camada, o valor é multiplicado por um peso que ajuda a definir o quanto o neurônio deve ser levado em consideração e depois é feito a soma com o desvio que aumenta ou diminui o valor influenciando o seu uso na função de ativação.

Essencialmente as redes neurais possuem dois fluxos diferentes, *backpropagation* (retropropagação) que faz o caminho inverso quando a função de perda tem como resultado um valor alto e *forward* propagation (propagação) que é o caminho “natural” que começa na camada de entrada e termina na função de perda (Figura 9).

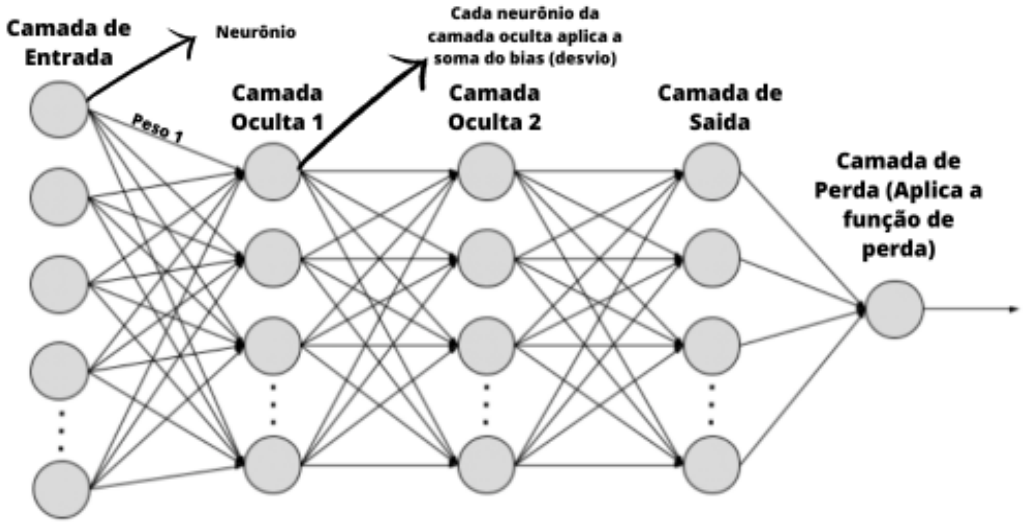

Figura 9 - Fluxo de propagação em uma rede neural.
Fonte: Adaptado de AWS.

Os valores de peso e desvio são muito importantes para o aprendizado da rede, pois estes são modificados e ajustados ao longo do treino para que a rede consiga chegar em resultados de saída melhores. Estes valores em um primeiro momento são gerados aleatoriamente e então passam pelas camadas até chegar à camada de perda, onde é aplicada a *loss function* (função de perda) que de maneira simplificada calcula o quão distante a predição feita pela rede está do resultado correto (Suzuki, 2011). Após ser feito o fluxo de propagação e ter o resultado do cálculo de erro, o algoritmo então aplica a retropropagação fazendo o caminho inverso atualizando os pesos e desvios, começando entre a camada de saída e a primeira camada oculta, até chegar novamente à camada de entrada (Figura 10).

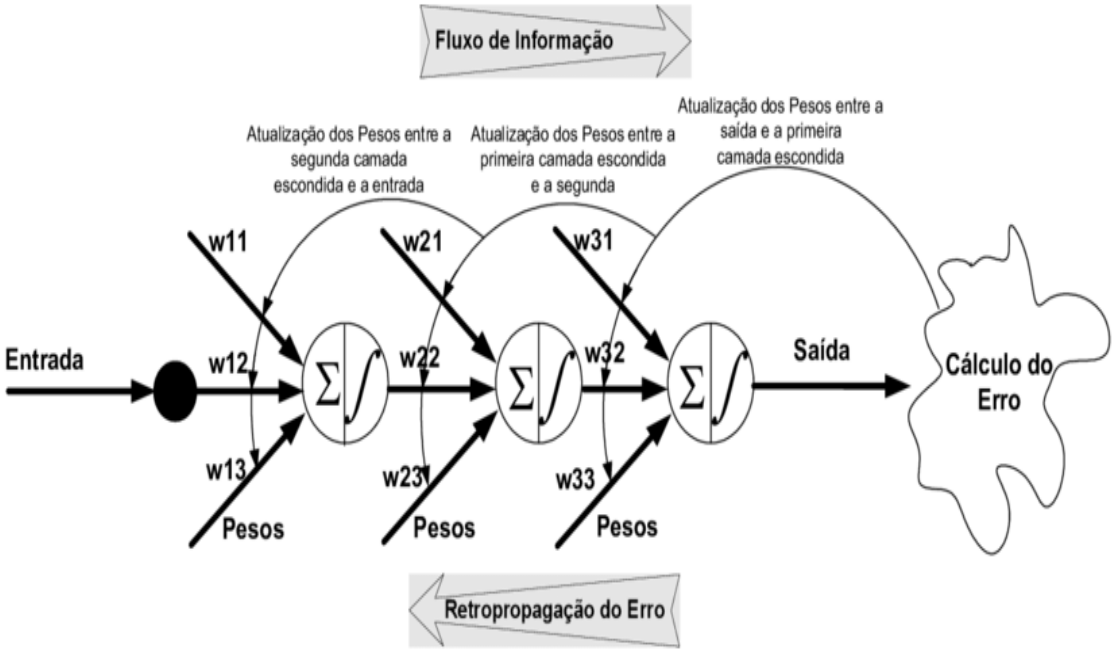

Figura 10 - Retropropagação em redes neurais.
Fonte: Barbosa, 2003.

No processo da retropropagação existem algumas técnicas utilizadas para fazer a otimização dos pesos e desvios, elas normalmente são baseadas no método gradiente descendente (Figura 11). Algumas das principais técnicas são *Stochastic Gradient Descent* (SGD) que altera os pesos após cada etapa de treino em direção ao gradiente negativo, Momentum que utiliza a média móvel dos gradientes para atualizar os pesos e *Root Mean Squared Propagation* (RMSProp) onde a taxa de aprendizado é adaptada para cada peso baseando-se na média de magnitudes dos gradientes recentes do peso (Ruder, 2016).

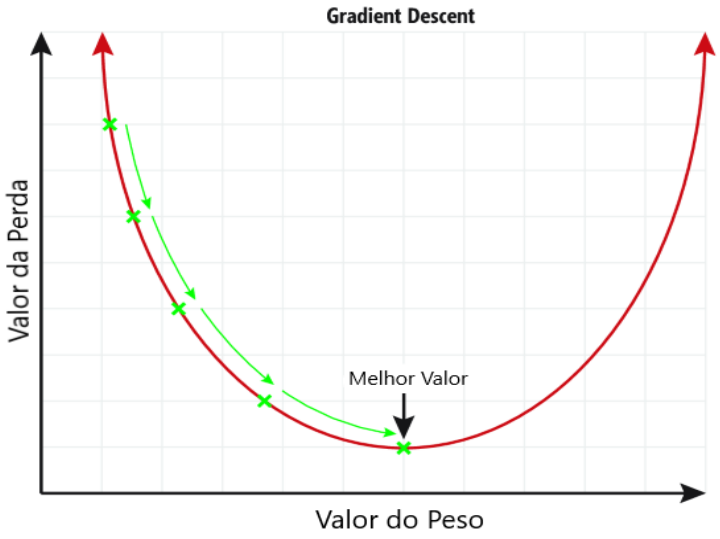

Figura 11 - Gradiente Descendente.
Fonte: Adaptado de LA, F., 2019.

Para entender o processo de aprendizado das redes neurais como um todo, também é importante saber o que são as funções de ativação. Essas funções são usadas em redes neurais para introduzir a não linearidade na saída de um neurônio. Elas são importantes porque permitem que as redes neurais modelem relações não lineares complexas entre entradas e saídas. Há muitas funções de ativação disponíveis para uso em redes neurais, algumas das mais populares são Linear, *Logistic Sigmoid, Tanh, ReLU, ELU* e *Softmax* (Dubey et al., 2022). É possível perceber que o valor vai ficar em uma escala restrita definida pela função, sendo este diferente dependendo da função utilizada (Figura 12).

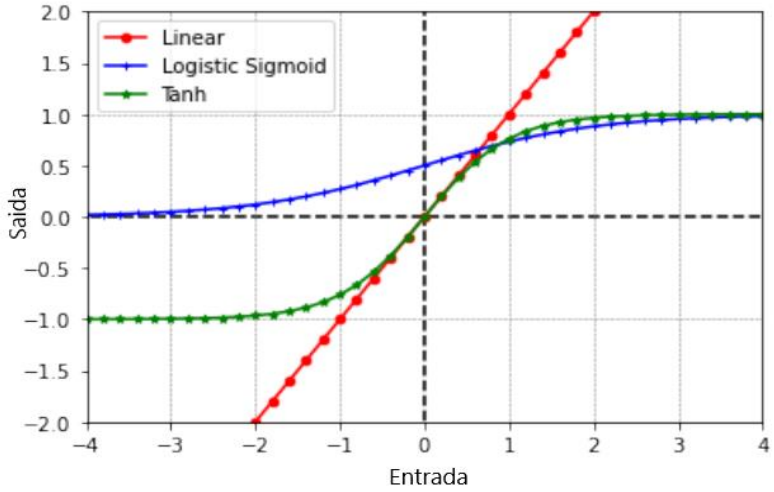

Figura 12 - Diferentes funções de ativação.
Fonte: Adaptado de Dubey et al., 2022.

As funções de ativação são utilizadas tanto nas camadas ocultas quanto na camada de saída, porém normalmente as que são usadas para a camada de saída são diferentes das que são utilizadas na camada oculta e são dependentes do tipo de predição requerido pelo modelo (Dubey et al., 2022).

O processo visto anteriormente onde ocorrem a propagação e a retropropagação é considerado uma iteração na rede neural, porém para que o algoritmo tenha um ganho de aprendizado alto, ela passa por várias épocas. Uma época pode ser definida como um ciclo onde todos os dados de entrada são utilizados exatamente uma vez, e pode ser composta por um ou mais lotes. Os lotes são uma forma de lidar com volume de dados muito grande, evitando sobrecarga na memória do computador (Dhande & Shaikh, 2019).

O tamanho do lote interfere diretamente em quantas iterações são feitas para cada época, caso o tamanho do lote seja menor que o tamanho total de pontos de

dados, será necessário fazer mais de uma iteração por época para passar por todos os pontos de dados (Figura 13).

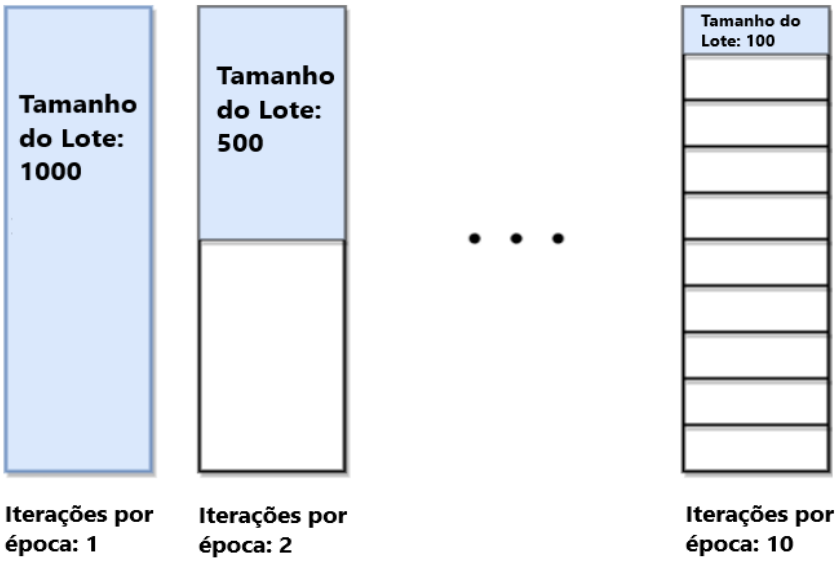

Figura 13 - Iterações, lotes e épocas.
Fonte: Adaptado de Baeldung.

A quantidade de épocas depende do tipo de problema que está sendo resolvido com o algoritmo, devem ser feitos testes e adaptações até encontrar um valor apropriado, onde não ocorra *underfitting* ou *overfitting* (Dhande & Shaikh, 2019).

O algoritmo de rede neural é consideravelmente mais complexo que outros similares, porém sua capacidade de aprendizado é muito maior, principalmente pela possibilidade de adaptação durante as diversas iterações que são feitas.

2.3 Bibliotecas de Machine learning

Como os algoritmos de *machine learning* são baseados em mátematica e estátisca em conjunto com a lógica de como são aplicados para serem capazes de aprender e prever resultados, eles podem ser implementados em praticamente qualquer linguagem de programação, porém esse seria um processo muito lento e custoso. Sendo assim foram criadas diversas bibliotecas que trazem as implementações dos algoritmos prontas para o uso, fácilitando o processo de criar um novo modelo de predição de *machine learning* (Raschka & Mirjalili, 2019).

A maioria dessas bibliotecas foram criadas para serem utilizadas com a linguagem *Python* que é uma linguagem de fácil aprendizado, principalmente por

possuír simplicidade na sintaxe e na manipulação dos dados (Raschka & Mirjalili, 2019).

Existem diversas bibliotecas com propósitos diferentes, como por exemplo: *Pandas* e *Numpy* que facilitam a etapa de pré-processamento dos dados, *Seaborn* que traz ferramentas para visualização dos dados em gráficos e outras como *Scikit-learn* e *Tensorflow* que são bibliotecas mais robustas e tem foco na implementação de algoritmos de *machine learning* (Rao et al., 2020).

2.3.1 Scikit-learn

O *Scikit-learn* é uma biblioteca de *machine learning* muito utilizada que possui código aberto e foi desenvolvida para a linguagem *Python*. Ela se tornou uma escolha popular por ser fácil de utilizar e possuir uma ótima documentação (Buitinck et al., 2013).

Além da sua práticidade, *Scikit-learn* prôve diversas funcionalidades para a etapa de pré-processamento e também uma vasta coleção de algoritmos de *machine learning* como por exemplo SVM, *Decision Tree*, KNN dentre outros que são implementados de forma eficiente e otimizados para *datasets* de larga escala (Pedregosa et al., 2011).

2.3.2 Tensorflow

Desenvolvido pela Google, *Tensorflow* é uma biblioteca de *machine learning* que possui um conjunto robusto de ferramentas para a criação e distrubuição de modelos, principalmente baseados em redes neurais, sendo muito utilizado para tarefas de aprendizado profundo (redes neurais com muitas camadas), aprendizado por reforço, processamento de linguagem natural e reconhecimento de imagens (Abadi et al., 2016).

O *TensorFlow* facilita a criação de modelos complexos com ferramentas de visualização, depuração e também com a API Keras que traz um grande nivel de abstração na etapa de desenvolvimento. A biblioteca também oferece uma arquitetura escalavel, possuindo ferramentas e bibliotecas para facilitar a implantação dos modelos de *machine learning* tanto em ambientes com servidores *web* quanto em dispositivos móveis e sistemas embarcados (Ramsundar et al., 2019).

2.4 Plataforma de Defesa Sanitária Animal do Rio Grande do Sul (PDSA-RS)

O setor avícola ocupa a terceira posição na cesta de produtos de exportação do agronegócio brasileiro, sendo muito importante para o país, tanto para comercialização interna quanto para a exportação, já que desde 2004 o Brasil lidera o *ranking* mundial de exportações de carne de frango (Silva, 2011).

Com o fato de que o país possuí produção de aves em grande escala, vários problemas relacionados a disseminação de doenças avícolas podem acontecer. Para evitar isso, existem alguns processos que devem ser realizados pelas granjas. Um dos principais é a certificação da granja, que possui diversas etapas a serem realizadas e tem como objetivo principal garantir que as granjas cumpram todas as normas exigidas pela lei, evitando assim a disseminação de doenças (Brasil, 2014).

Os processos para a garantia de que os estabelecimentos avícolas estejam operando de acordo com a legislação são complexos e envolvem algumas entidades, sendo essas: (a) Responsavel(eis) pelo estabelecimento avícola, (b) Serviço Veterinário Estadual (SVE), (c) Laboratório Agropecuário (Laboratório) (d) Ministério da Agricultura, Pecuária e Abastecimento (MAPA).

A plataforma do PDSA-RS tem como objetivo sistematizar a cadeia de operações feitas dentre essas entidades. De acordo com Freitas (2023, p. 36),

> A arquitetura da PDSA-RS é composta por portais, os quais são acessados pelos principais atores existentes na área de sanidade animal, cada portal composto por módulos que implementam funcionalidades (tarefas) do processo ao qual aquele módulo foi implementado. Os portais são: (a) Portal do Serviço Veterinário do Estado do Rio Grande do Sul (SVE), (b) Portal do Responsável Técnico (RTs), (c) Portal do Laboratório Agropecuário (Laboratório) (d) Portal do Ministério da Agricultura (MAPA) (e) Portal da Administração (Admin);

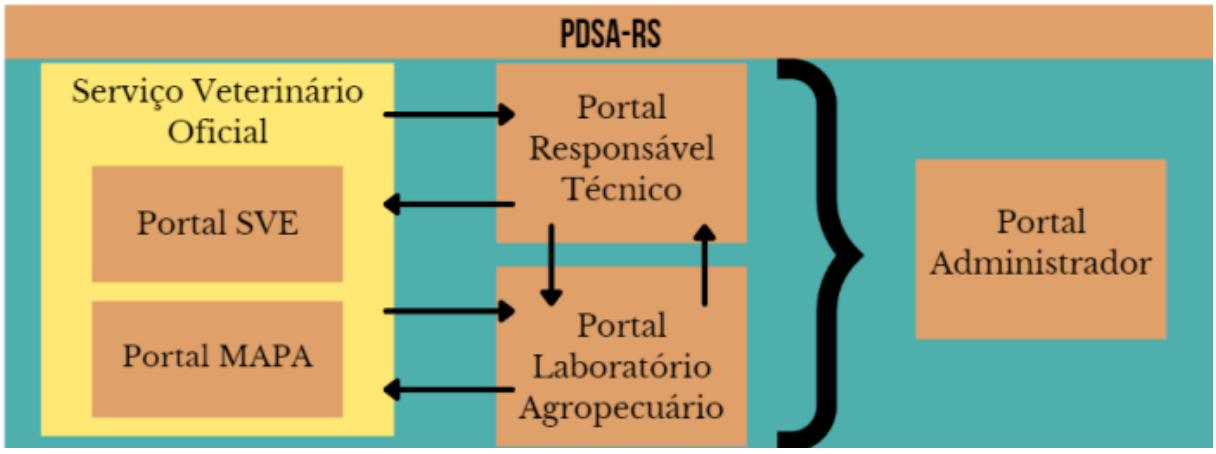

Figura 14 - Relacionamento entre os portais do PDSA-RS
Fonte: Freitas, 2023.

Na Figura 14, é possivel visualizar o relacionamento entre as entidades através dos portais existentes na plataforma do PDSA-RS.

Existem diversos módulos na plataforma, incluindo o módulo de certificação de granjas avícolas, onde as etapas do processo são registradas de forma digital. Sendo assim a PDSA-RS traz muitos benefincios, tanto para produtores quanto para entidades oficiais do governo, de maneira que facilita e agiliza o processo de controle da sanidade animal.

3 ESTUDO DE CASO

O estudo de caso foi desenvolvido com foco na aplicação de *machine learning* para auxiliar usuários do PDSA-RS na etapa da coleta de materiais que faz parte do processo de certificação das granjas avícolas.

No processo de certificação de aves, as granjas passam por alguns procedimentos, um destes é chamado de monitoria, que se resume em fazer a coleta de materiais passíveis de serem analisados para a detecção de alguma doença, esses materiais são coletados dos aviários, acondicionados em recipientes lacrados e enviados para um laboratório agropecuário que por sua vez faz análises à procura de doenças (Brasil, 2020). Existem diversas regras para quais materiais devem ser coletados na monitoria, dependendo de diversos fatores, mas principalmente a idade das aves e o tipo de estabelecimento (Brasil, 2020).

Esse estudo foi feito com o objetivo de gerar um modelo capaz de classificar quais materiais devem ser coletados em uma monitoria, com base em alguns parâmetros (*features*) como por exemplo a idade das aves, o tipo de exploração da granja, vacinas que foram aplicadas nas aves, dentre outros.

Existem diversos materiais que podem ser coletados, porém a coleta de alguns conjuntos de materiais específicos se repete muitas vezes, a ideia foi encontrar a correlação das *features* com estes conjuntos de materiais que possuem um grande número de ocorrências para gerar um modelo para classificar qual grupo de materiais deve ser coletado a partir de alguns parâmetros.

As principais ferramentas utilizadas foram a linguagem *Python* em conjunto com a biblioteca *Scikit-learn* que possui uma vasta lista de ferramentas que auxiliam a aplicação de algoritmos de *machine learning*. Para a implementação do algoritmo de redes neurais foi utilizada a biblioteca *TensorFlow* em conjunto com a API (Interface de Programação de Aplicação) Keras que fornece uma interface de alto nível,

facilitando o uso do *TensorFlow*. Também foram utilizadas as bibliotecas *Pandas* e *Numpy* para auxiliar na manipulação dos dados e a biblioteca *Seaborn* para visualização gráfica.

3.1 Pré-processamento, implementação e resultados

A extração dos dados foi feita em uma base de dados fornecida pelo PDSA-RS com comandos SQL (Linguagem de Consulta Estruturada) e em seguida foi gerado um arquivo CSV a partir dos resultados da consulta. O *dataset* foi extraído do banco de dados com os seguintes campos:

Finalidade	Finalidade da granja, ex: granja de produção, granja de ciclo completo...
Idade	Idade das aves, (campo em formato de texto com um número e a unidade de medida de tempo em dias ou semanas).
Materiais coletados	Lista dos materiais coletados na monitoria.
Número de aves	Número de aves no núcleo em que a coleta foi realizada.
Tipo de exploração	Exploração no núcleo referente a linhagem das aves, ex: matriz, avó...
Vacina Newcastle	Status da vacina de Newcastle em relação às aves em que a monitoria foi feita.
Vacina Salmonella	Status da vacina de Salmonella em relação às aves em que a monitoria foi feita.

Fonte: Autor (2023).

A primeira etapa foi transformar a idade para um campo inteiro contendo o valor em dias, depois, foi necessário fazer o encoding (codificação de variáveis do tipo string (texto) para números) de todos os dados categóricos para possibilitar a aplicação dos

algoritmos, além disso, os materiais coletados são conjuntos de itens, porém estavam no formato de string, logo, foi necessário fazer a transformação para facilitar a manipulação do *dataset*. A etapa mais crítica do pré-processamento foi a redução do número de classes, pois existiam muitas classes pelo fato de algumas ocorrências possuírem um conjunto de materiais incomum que foram considerados como outliers, esses dados foram removidos diminuindo o número total de classes para 3, sendo estas (a) Mecônio e ovos bicados. (b) Soros e propé misto. (c) Soros e fezes frescas.

Os primeiros resultados obtidos, utilizando os hiperparâmetros padrões do *Scikit-learn* foram os seguintes: KNN com 56% de acurácia, *Decision Tree* com 74.94%, SVM com 58.58% e 83% de acurácia com redes neurais. Com estes resultados foi possível perceber que apenas os algoritmos *Decision Tree* e redes neurais tiveram uma precisão razoável, então com a realização de alguns testes notou-se que os campos número de aves e finalidade apresentavam uma correlação muito baixa, atrapalhando o aprendizado dos algoritmos.

A idade é a *feature* de maior importância, então foi analisado a correlação de outros campos com a idade em busca de encontrar quais melhoram os resultados de predição, por fim as *features* utilizadas foram as seguintes: idade, tipo de exploração (Figura 15), vacina *newcastle* (Figura 16) e vacina salmonella (Figura 17) e as outra *features* foram removidas.

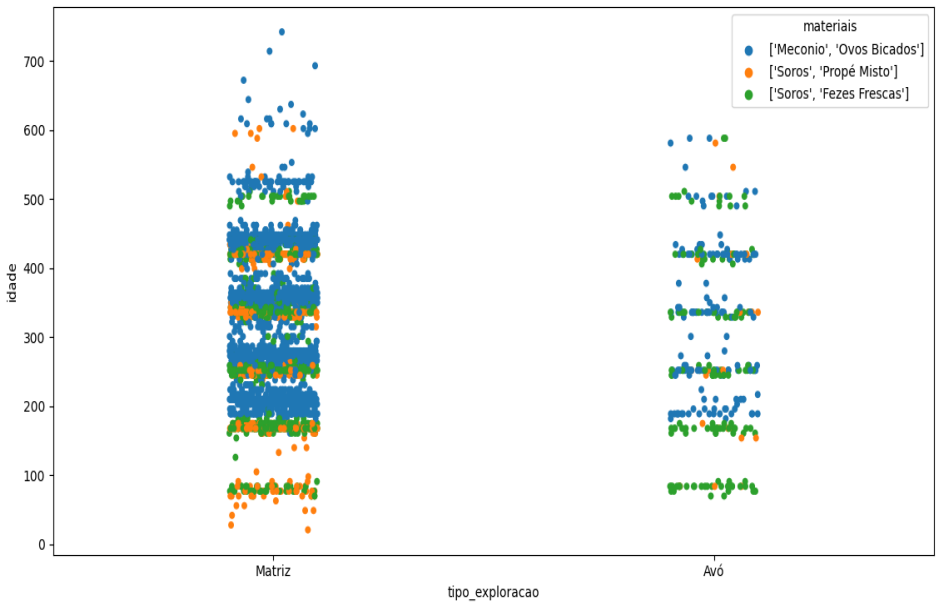

Figura 15 - Correlação de idade e tipo de exploração.
Fonte: Autor (2023).

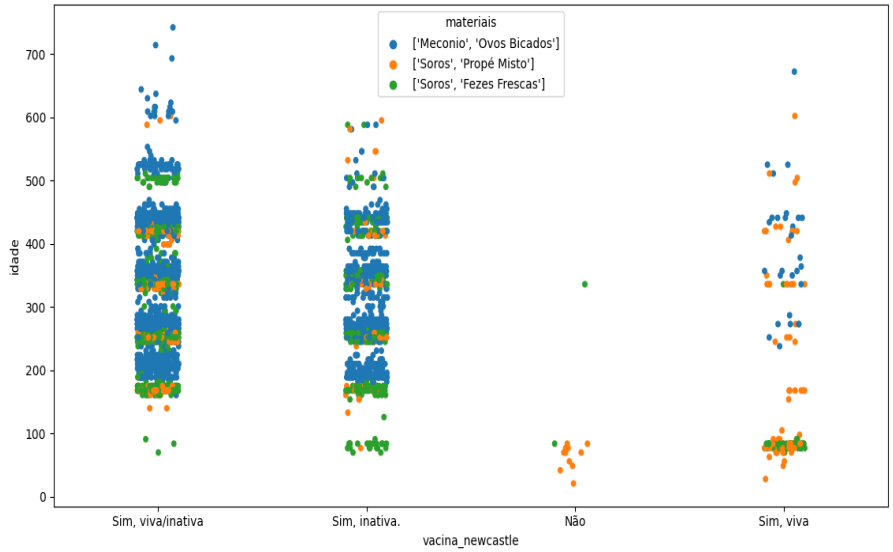

Figura 16 - Correlação de idade e vacina *newcastle*.
Fonte: Autor (2023).

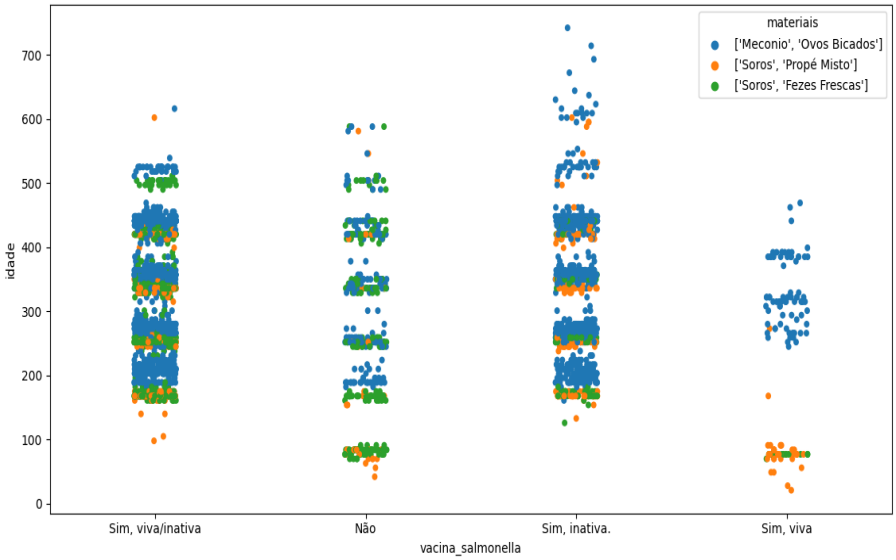

Figura 17 - Correlação de idade e vacina Salmonella.
Fonte: Autor (2023).

Para o processo de *tuning* foi utilizado o módulo *Grid Search* CV da biblioteca *Scikit-learn* (Scikit-learn, 2023), que faz uma busca exaustiva procurando os parâmetros com melhores resultados.

Dos algoritmos tradicionais, o com maior acurácia foi o KNN com 79.88% utilizando 26 como número de vizinhos, distância como peso e a métrica de manhattan, já o *Decision Tree* chegou a 78.94% utilizando como critério o índice de gini, profundidade máxima 10 e o máximo de *features* calculado por log2. O SVM que teve o pior resultado antes dos ajustes, conseguiu uma melhoria significativa,

principalmente pela mudança do kernel linear para o kernel RBF (Radial Basis Function Kernel) e obteve 78% de taxa de acerto com gamma 0.2 e C (regularização) 2.

Já com redes neurais a acurácia final foi de 94%. Para chegar neste resultado foi criado apenas uma camada oculta, a função de ativação utilizada para a camada de entrada e para a camada oculta foi a ReLu e para a camada de saída a função Logistic Sigmoid. Para otimização foi utilizado o algoritmo Adam, para o cálculo de perda foi utilizado o método do Keras *sparse categorical cross entropy* e a métrica de acurácia categórica e por fim no treino foram utilizadas 12 épocas, valor que foi definido analisando o momento em que a diminuição do valor de perda começou a ficar muito baixo.

4 CONCLUSÃO

O foco do trabalho foi em técnicas de classificação, tendo uma abordagem relativamente rasa sobre o assunto, demonstrando assim, o quão grande é a área de *machine learning*. Também foi possível perceber que a aplicação dessas técnicas pode gerar oportunidade de pesquisa em muitas outras áreas de estudo pelo fato de que com a existência de dados reais e com a adaptação desses dados se faz possível criar estudos com diversas finalidades utilizando técnicas de *machine learning*.

A etapa do referencial teórico foi muito importante para adquirir conhecimento e ter um entendimento maior de como as técnicas de *machine learning* são utilizadas e principalmente como os algoritmos funcionam, fazendo assim com que os conhecimentos adquiridos pudessem ser utilizados no estudo de caso deste trabalho.

No primeiro momento do estudo de caso, onde não foram feitos ajustes nas *features* nem nos hiperparâmetros, para os algoritmos tradicionais foi possível notar que o algoritmo *Decision Tree* teve uma performance superior aos demais algoritmos. Após o processo de *tuning* e melhoria na utilização das *features*, o *Decision Tree* não teve muita melhora, já os algoritmos SVM e KNN melhoraram muito em comparação aos resultados anteriores. Mesmo assim o algoritmo com melhor resultado foi o KNN com 79.88% de acurácia, sendo este um resultado não muito bom para um classificador. Já com redes neurais os resultados foram muito melhores obtendo 83% antes do processo de *tuning* e 94% após, mostrando que apesar de serem mais complexos estes algoritmos possuem uma capacidade de aprendizado muito grande.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M. et al. TensorFlow: A System for Large-Scale Machine Learning. *In*: 12th USENIX Symposium on Operating Systems Design and Implementation, v. 16, p. 265-283, ISBN 978-1-931971-33-1, 2016.

AKPAN, U. I. & STARKEY, A. Review of classification algorithms with changing inter-class distances. *Machine Learning with Applications*, v. 4, p. 100031, 2021.

AWS. O que é uma rede neural? Disponível em: https://aws.amazon.com/pt/what-is/neural-network/. Acesso em: 03 de junho de 2023.

BARBOSA, L.F.F.P.W. **Estudo de um sistema inteligente para o controle de posição do plasma no Tokamak - ETE**. 2003. Tese (Doutorado em Engenharia Eletrônica e Computação) - Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, 2003.

BONACCORSO, G. *Machine learning* algorithms. *Packt Publishing Ltd*, 2017. E-*book* (353 p.). ISBN 978-1-78588-962-2.

BRASIL. Instrução Normativa Nº 21, de 21 de outubro de 2014. Ministério da Agricultura, Pecuária e Abastecimento: Secretaria de Defesa Agropecuária, 2014.

BRASIL. Ministério da Agricultura, Pecuária e Abastecimento. Secretaria de Defesa Agropecuaria. Departamento de Saude Animal. OFÍCIO-CIRCULAR Nº 82/2020/DSA/SDA/MAPA. Brasília, 19 de outubro de 2020. Assunto: Procedimentos sanitários adicionais aplicados na importação de material genético avícola - MGA.

BUITINCK, L. et al. API Design for Machine Learning Software: Experiences from the Scikit-learn Project. *In:* ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013.

CUNNINGHAM, P. et al. Supervised learning. In: CORD, M. & CUNNINGHAM, P. *Machine Learning Techniques for Multimedia*. 2008, p. 21-49.

DHANDE, G. & SHAIKH, Z.S. Analysis of Epochs in Environment based Neural Networks Speech Recognition System. 3. ed. *International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, p. 605-608, 2019.

DUBEY, S.R. et al. Activation Functions in Deep Learning: A Comprehensive Survey dubey and Benchmark. *Neurocomputing*, v. 503, p. 92-108, 2022.

FREITAS, G. D. Metodologia de modelagem e implementação de blockchain em processos de negócio. 2023. 56p. Dissertação (Mestrado em Ciência da Computação), - Universidade Federal de Santa Maria, Santa Maria, RS, 2023.

GLIELMO, A. et al. Unsupervised learning methods for molecular simulation data. *Chemical Reviews,* v. 121, n. 16, p. 9722−9758, 2021.

JAIN, V. Introduction to KNN Algorithms. **Analytics Vidhya,** 2022. Disponível em: https://www.analyticsvidhya.com/blog/. Acesso em: 28 de novembro de 2022.

JIJO, B.T. & ABDULAZEEZ, A.M. Classification Based on Decision Tree Algorithm for *Machine Learning*. *The Journal of Applied Science and Technology Trends*, v. 2, n. 1, p. 20-28, 2021.

KIM, E. Everything You Wanted to Know about the Kernel Trick. **Erick Kim**, 2015. Disponível em: http://www.eric-kim.net/eric-kim-net. Acesso em: 3 de dezembro de 2022.

K-Nearest Neighbor (KNN) Algorithm for Machine Learning. **JavaTpoint***,* 2021. Disponível em: <https://www.javatpoint.com/>. Acesso em: 24 de novembro de 2022.

LA, F. How do neural networks learn?. *MSDN Magazine*, v. 34, n. 4, 2019.

LECUN, Y. et al. Deep learning. *Nature*, v. 521. p. 436-444, 2015.

LORENA, A.C & CARVALHO, A.C.P.L.F. Uma introdução às *Support Vector Machines. Revista de Informática Teórica e Aplicada,* v. 14, n. 2, 2007.

LUDERMIR, T.B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. Estudos avançados, v. 35, n. 101, p. 85-94, 2021.

Machine Learning Glossary. **Google Developers**, 2022. Disponível em: <developers.google.com>. Acesso em: 15 de janeiro de 2023.

MAHESH, B. *Machine learning* algorithms - a review. *International Journal of Science and Research*, v. 9, n. 1, p. 381-386, 2020.

OSISANWO, F.Y. et al. Supervised machine learning algorithms: classification and comparison*. International Journal of Computer Trends and Technology,* v. 48, n. 3, p. 128-138, 2017.

PAYETTE & LU. Artificial Intelligence and Machine Learning *in*: CS106E Spring, 2018.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825-2830, 2011.

PODGORELEC, V. et al. Decision Trees: An Overview and Their Use in Medicine. *Journal of Medical Systems,* v. 26, n. 5, p. 445-463, 2002.

PRIYA, D.S.V.S. et al*.* A Review on the Importance of Machine Learning and Artificial Intelligence in Real Life Problem Solving. *Journal of Computational and Theoretical Nanoscience*, v. 17, n. 9, p. 4336-4339, 2020.

RAMSUNDAR, B. et al. TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning. *O'Reilly Media*, 2019.

RAO, S. et al. A comparative study between various preprocessing techniques for Machine Learning. *International Journal of Engineering Applied Sciences and Technology*, v. 5, n. 3, p. 431-438, 2020.

RASCHKA, S., & MIRJALILI, V. Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow 2. Packt Publishing, 2019.

RUDER, S. An overview of gradient descent optimization algorithms*. arXiv*, 2016. Disponível em: <https: //arxiv.org/abs/1609.04747>. Acesso em: 25 de maio de 2023. arXiv:1609.04747.

SCIKIT-LEARN. *GridSearchCV*. 2023. Disponível em: <https://scikit-learn.org/stable/index.html>. Acesso em: 06 de julho de 2023.

SHALEV-SHWARTZ, S., & BEN-DAVID, S. Understanding Machine Learning: From Theory to Algorithms. *Cambridge University Press*, ISBN978-1-107-05713-5, 2014.

SHAMRAT, F.M.J.M. et al. Performance evaluation among ID3, C4.5, and CART decision tree algorithm. In: RANGANATHAN, G. et al. *Pervasive Computing and Social Networking*, v. 317, p. 127-142, 2021.

SILVA, M.A.P. et al. Oferta de exportação de carne de frango do Brasil, de 1992 a 2007. *Revista* de *Economia* e *Sociologia Rural*, v. 49, n. 1 p. 31-54, 2011.

SINGH, S. & GUPTA, P. Comparative study id3, CART and C4.5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology*, v. 27, n. 27, p. 97-103, 2014.

SOOFI, A.A. & AWAN, A. *Classification techniques in machine learning: applications and issues*. J*ournal of Basic & Applied Sciences*, v. 13, p. 459-465, 2017.

Support Vector Machines. **Scikit-learn.** Disponível em: https://scikit-learn.org/. Acesso em: 23 de dezembro de 2022.

SUTTON, R.S. & BARTO, A.G. Reinforcement learning: an introduction. *The MIT Press,* 2018. *E-book* (352 p.). ISBN 978-02-6203-924-6.

SUZUKI, K. Artificial Neural Networks - Methodological Advances and Biomedical Applications. 1. ed. Croatia: InTech, 2011. 362 p. ISBN 978-953-307-243-2.

TAUNK, K. et al. A brief review of nearest neighbor algorithm for learning and classification. *International Conference on Intelligent Computing and Control Systems (ICCS)*, p. 1255-1260, 2019.

Tuning an SVM Model. **IBM***,* 2021. Disponível em: https://www.ibm.com/br-pt. Acesso em: 28 de outubro de 2022.