UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Algoritmos de aprendizado de máquina supervisionado para classificação do comportamento de rebanhos de vacas

Matheus Paulo dos S. Demiro

Recife - PE
2019

Matheus Paulo dos S. Demiro

Algoritmos de aprendizado de máquina supervisionado para classificação do comportamento de rebanhos de vacas

Artigo científico desenvolvido na disciplina de Reconhecimento de Padrões apresentado na Universidade Federal Rural de Pernambuco como requisito básico para avaliação curricular do curso de Sistemas de Informação

Orientador(a): Rodrigo Soares

Recife - PE
2019

RESUMO

Neste artigo é realizado uma análise comparativa entre algoritmos de aprendizagem de máquina para classificação do comportamento de rebanhos de vacas através de dados relacionados a postura do animal em pastejo e do ambiente em seu entorno. Os dados considerados para o desenvolvimento deste trabalho foram as variáveis meteorológicas (medidas de temperatura) coletadas por Pedro Henrique Dias Batista, estudante do Programa de Pós-graduação em Engenharia Agrícola da UFRPE (Universidade Federal Rural de Pernambuco). O principal objetivo deste artigo é realizar análise comparativa de desempenho entre algoritmos de aprendizado de máquina supervisionado para classificar o comportamento das vacas em pastejo. Os classificadores considerados para este artigo foram: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes Learning, Árvores de Decisão, Redes Neurais Artificiais e Random Forest. Esses algoritmos de classificação determina se o animal está ruminando, comendo, andando, em ócio ou bebendo, contribuindo dessa forma no processo de monitoramento e manejo do rebanho, além do apoio a tomada de decisão dos pecuaristas. Após as análises comparativas, considerando o f-measure (em portugês, medida F) como métrica avaliativa, concluiu-se que o Random Forest apresentou um melhor desempenho em relação aos demais classificadores.

Palavras-chave: Previsão de comportamento animal. Comportamento das vacas. Manejo de rebanho. Monitoramento de rebanho.

ABSTRACT

In this paper, a comparative analysis was performed between machine learning algorithms to classify the behavior of cow herds through data related to the posture of the grazing animal and the environment in its surroundings. The data considered for the development of this work were the meteorological variables (temperature measurements) collected by Pedro Henrique Dias Batista, student of the Post-graduate Program in Agricultural Engineering of UFRPE (Federal Rural University of Pernambuco). The main objective of this article is to perform a comparative performance analysis among supervised machine learning algorithms to classify cow grazing behavior. The classifiers considered for this paper were: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes Learning, Decision Trees, Artificial Neural Networks and Random Forest. These classification algorithms determine whether the animal is ruminating, eating, walking, leisure or drinking, thus contributing to the process of monitoring and managing the herd, in addition to support for the decision-making of cattle ranchers. After the comparative analyzes, considering f-measure (in Portuguese, measure F) as an evaluation metric, it was concluded that Random Forest presented a better performance in relation to the other classifiers.

Keywords: Prediction of animal behavior. Behavior of cows. Herd management. Herd monitoring.

1 INTRODUÇÃO

No Brasil, uma cadeia produtiva de destaque é a pecuária leiteira com grande importância como atividade econômica e social, sendo contemplada nos últimos anos com ferramentas tecnológicas que impulsionaram o desenvolvimento e surgimento de novas técnicas e políticas relacionados a pecuária leiteira (ERVILHA, 2014). A tecnologia surge como nova aliada para os pecuaristas, provocando a redução de mão de obra nos sistemas de produção de leite e, consequentemente, com a redução acentuada dos custos de produção. Essas soluções tecnológicas se estendem desde dispositivos IoT a chips utilizados para coleta e processamento de dados individuais das vacas em tempo real.

O crescimento acentuado na pecuária leiteira é caracterizado pelo aumento do tamanho de rebanhos, decorrente do marcante aumento de produtividade e pela redução de mão de obra especializada nos sistemas de produção. Além disso, a criação de vacas no Brasil que fomenta a atividade leiteira apresenta características próprias, sendo carente em mão de obra especializada, com trabalho familiar e pouca disponibilidade de recursos (ERVILHA, Gabriel Teixeira). Majoritariamente as tecnologias ofertadas pelo mercado possuem preços elevados que impossibilitam o acesso dos pequenos pecuaristas (criadores de vacas) a essas ferramentas.

Com a limitação de recursos e carência de mão de obra especializada, outro fator pertinente é o processo de monitoramento individual dos animais. O maior desafio enfrentado pelos pecuaristas são as coletas manuais dos dados individualizados do rebanho. Portanto, existe uma necessidade de implantar não só sensores capazes de identificar e gerar dados individuais dos animais, mas também de sistemas que permitam a interpretação desses dados para transformar em informações úteis e estratégicas em apoio à tomada de decisão do pecuarista.

Baseando-se nessa necessidade, o projeto em questão apresenta algoritmos de aprendizado de máquina supervisionado (AMS) que são capazes de interpretar os dados coletados do rebanho e classificá-los em um dos cinco possíveis estados, sendo eles: ruminando, comendo, bebendo, ócio e andando. Os algoritmos AMS classificadores utilizados neste trabalho foram: Naive Bayes, árvore de decisão, redes neurais, SVM e KNN. Com isso, o principal objetivo deste projeto é realizar uma análise comparativa entre esses algoritmos e apresentar os resultados obtidos no processo de classificação dos dados. Além disso, também tem-se o objetivo de contribuir no controle e monitoramento das vacas de determinado rebanho, auxiliando o pecuarista no manejo e tomada de decisão.

Os dados utilizados neste projeto envolvem variáveis meteorológicas, sendo elas: "tbs"(temperatura de bulbo seco - é temperatura do ar), "ur"(umidade relativa - quantidade de água presente no ar em relação ao seu ponto de saturação), "tgn"(tem-peratura de globo negro - indica efeitos combinados da energia radiante, temperatura e velocidade do ar) e "tpo"(tem-peratura de orvalho - é a temperatura a qual o vapor de água que está em suspensão no ar começa a se condensar). Estas variáveis interferem no comportamento dos animais, principalmente em regiões de altas temperaturas onde o desconforto térmico nos pastos causam estresse nos bovinos

de leite. Isso desencadeia variações nos padrões de comportamento do bovino, com redução no tempo de alimentação e ruminação e, aumento no tempo de ócio, numa provável tentativa de diminuir a produção de calor metabólico (GERON, 2014).

2 TRABALHOS RELACIONADOS

Existem diversas tecnologias e sistemas computacionais que utilizam análises comportamentais de determinado rebanho para realizar o monitoramento e controle do estresse de cada animal. Para isso, é necessário coletar, processar e analisar os dados relativos a rotina diária do rebanho. Logo, nota-se a necessidade de utilizar um sistema capaz de aprender, analisar grande volume de dados e tomar rápidas decisões autônomas. Por isso, a maioria dos trabalhos já desenvolvidos estão relacionados com aprendizado de máquina e tecnologias IoT de baixo custo.

Luis et al. (2018) propôs uma plataforma de monitoramento de comportamento animal, baseado em tecnologias IoT. A proposta inclui uma rede local IoT para reunir os dados de localização geográfica, coletados pelos colares eletrônicos individuais utilizados pelos animais, e uma plataforma na nuvem, com capacidades de processamento e armazenamento, para autonomamente pastorear ovinos dentro de vinhas. A plataforma também incorpora recursos de aprendizagem de máquina (AM), permitindo a extração de informações relevantes dos dados coletados pelo colar eletrônico. Foram avaliados diversos algoritmos AM para testar a precisão na detecção das infrações posturais de ovelhas, sendo eles: Random Forest ("Floresta Aleatória"), Decision Trees ("Árvores de Decisão") usando pacotes C50 e rpart, XGBoost, K-Nearest Neighbor (KNN), Support Vector Machine (SVM) e Naive Bayes. Como resultado desse experimento, todos os algoritmos apresentaram acurácia semelhante, porém os resultados obtidos utilizando árvore de decisão foram mais relevantes (NOBREGA, 2018).

O artigo realizado por Luis et al. objetiva a criação de uma plataforma IoT completa e autossuficiente, também apresenta uma breve seção de comparação entre os algoritmos de aprendizado de máquina selecionados. Esta seção, não descreve o processo de seleção dos modelos e não esclarece ao leitor de onde as taxas de acerto dos algoritmos foram retirados. Em outras palavras, omitiu-se parte da análise comparativa entre os algoritmos AM. Já neste trabalho, será realizada uma análise totalmente transparente e esclarecedora, sendo possível comprovar a eficiência da estimativa dos algoritmos.

Williams et al. (2016) apresenta um modelo de monitoramento comportamental da vaca leiteira à base de pasto através de dados GPS usando técnicas de mineração de dados e aprendizado de máquina. Esse trabalho tem como principal objetivo construir um modelo autônomo de monitoramento de rebanhos de vacas, utilizando dados do sistema de posicionamento global (GPS) coletados por colares ao redor do pescoço das vacas, superando os esforços das observações manuais. Para realizar a extração de características relevantes dos dados coletados optou por utilizar o poder computacional da aprendizagem de máquina. Para a construção de um

modelo robusto e preciso, foram avaliados diversos algoritmos AM para testar a precisão de classificação dos estados de pastagem, repouso e caminhada, sendo eles: Naive Bayes, JRip, J48 e Random Forest. Nesse caso, os resultados obtidos inferiu que o classificador JRip é o mais adequado para a implementação do modelo (WILLIAMS, 2016).

O estudo realizado por Williams et al. utiliza dados do GPS, coletados por meio de colares eletrônicos, para classificar os dados em três possíveis estados: pastagem, repouso e caminhada. A precisão dos resultados é relativamente baixa, especialmente para os estados de pastagem e repouso, devido ao uso de um único sensor que registra a posição do animal. Além disso, apresenta apenas a classificação de três possíveis estados de comportamento do animal, enquanto que o presente artigo classifica os dados em cinco possíveis estados: comendo, ócio, andando, bebendo e ruminando.

3 MÉTODOS

3.1 NAIVE BAYES LEARNING

O algoritmo Naive Bayes é provavelmente o classificador probabilístico mais utilizado em Machine Learning (aprendizado de máquina). O termo "Naive"(ingênuo) é devido a sua propriedade de assumir que os atributos são condicionalmente independentes, ou seja, a informação de um evento não é informativa sobre nenhum outro (OGURI, 2006). Já o nome "Bayes"está relacionado ao fato do modelo matemático probabilístico implementado pelo algoritmo ser baseado baseado no "Teorema de Bayes", desenvolvido pelo pastor protestante e matemático inglês Thomas Bayes no século XVII para tentar provar a existência de Deus (GONÇALVES, 2018).

Alguns autores afirmam que o raciocínio diagnóstico dos médicos são naturalmente bayesianos, ou seja, dado uma série histórica de ocorrências de determinados eventos relacionados, por exemplo, ao sintoma de dor no peito, é comum os clínicos imaginarem um conjunto de possibilidades diagnósticas. Logo, pode-se inferir que o raciocínio do médico está alinhado aos ideias da probabilidade condicional do Teorema de Bayes.

Probabilidade condicional, é a probabilidade de ocorrência de um evento A, dado que ocorreu o evento B, sendo representado por: P(A|B). Para o caso do aprendizado de máquina, deseja-se saber qual a probabilidade da classe ocorrer dada uma instância (evento que já ocorreu). Logo, o Teorema de Bayes pode ser derivado a partir da definição de probabilidade condicional, sendo:

$$P(\mathrm{A}|\mathrm{B}) = \frac{P(\mathrm{A} \cap B)}{P(B)}, \text{ se } \mathrm{P(B)} \neq 0. \tag{1}$$

$$P(B|\mathrm{A}) = \frac{P(B \cap \mathrm{A})}{P(\mathrm{A})}, \text{ se } \mathrm{P(A)} \neq 0. \tag{2}$$

$$P(B \cap \mathrm{A}) = P(B|\mathrm{A}) \times P(\mathrm{A}) \tag{3}$$

como a operação de interseção é simétrica, tem-se:

$$P(\mathrm{A} \cap B) = P(B \cap \mathrm{A}) \tag{4}$$

$$P(\mathrm{A}|\mathrm{B}) \times P(B) = P(B|\mathrm{A}) \times P(\mathrm{A}) \tag{5}$$

ajustando os termos, conclui-se o Teorema de Bayes em:

$$P(\mathrm{A}|B) = \frac{P(B|\mathrm{A}) \times P(\mathrm{A})}{P(B)}, \text{ se } \mathrm{P(B)} \neq 0. \tag{6}$$

O classificador Naive Bayes assume a premissa de que todos os atributos são independentes entre si. Ou seja, cada atributo A_i, por exemplo, influencia uma classe C, mas nenhum exerce influência sobre o outro. Isso é bastante irreal, visto que no processo de aprendizado de máquina é essencial um grau aceitável de correlação entre os atributos das instâncias. Porém, segundo estudos realizados por (ZHANG, 2004) essa premissa irreal torna-se pouco relevante na prática, pois o classificador Naive Bayes apresenta resultados comparáveis a outros algoritmos complexos (KOGA, 2011). Logo, assumindo essa premissa como verdadeira e adaptando o Teorema de Bayes ao contexto de AM, tem-se

$$P(C|\mathrm{A}_1, ..., \mathrm{A}_n) = \frac{P(C) \times \prod_{i=0}^{n} P(\mathrm{A}_i|C)}{P(\mathrm{A})}, \text{ se } \mathrm{P(A)} \neq 0. \tag{7}$$

A premissa da independência de atributos é o principal motivo por torná-lo bastante usual e comum em trabalhos que envolvam aprendizado de máquina. Além da simplicidade do modelo matemático, o algoritmo não precisa da estimação de parâmetros, pois apenas as probabilidades a priori de classe e as condicionais são suficientes (KOGA, 2011).

A estimativa de probabilidades é a tarefa desempenhada no processo de aprendizagem do Naive Bayes. Esta tarefa, calcula a probabilidade baseada na frequência relativa de cada classe (probabilidades a priori) e cada atributo dada a classe (probabilidades condicionais) no conjunto de treinamento. Ou seja, para calcular a classe mais provável da nova instância, calcula-se a probabilidade de todas as possíveis classes e, no fim, escolhe-se a classe com a maior probabilidade como rótulo da nova instância (PARDO, 2002). Portanto, tem-se a finalidade de maximizar o termo $P(C|\mathrm{A}_1, ..., \mathrm{A}_n)$ da equação (7). Para isso, é necessário maximizar $P(C) \times \prod_{i=0}^{n} P(\mathrm{A}_i|C)$ o denominador e minimizar o numerador $P(\mathrm{A})$. Assumindo a premissa de independência dos atributos, tem-se que o denominador é uma constante e pode-se anulá-lo da equação (7), resultando na fórmula:

$$\arg\max \mathrm{P}(C|\mathrm{A}_1, ..., \mathrm{A}_n) = \arg\max \mathrm{P}(C) \times \prod_{i=0}^{n} P(\mathrm{A}_i|C) \tag{8}$$

onde $P(C)$ é o número total de casos pertencentes a classe C dividido pelo número total de casos. Já $P(\mathrm{A}_i|C)$ é o número de casos da classe C com atributo A_i sobre o número total de

casos da classe.

Importante ressaltar, que é possível a ocorrência de casos em que um atributo A_i não se manifesta para determinada classe, logo, a probabilidade será zero. Pois, se uma determinada probabilidade assumir valor zero, o resultado do produtório da equação (8) será zero. Esse problema é chamado de "frequência-zero". Para solucioná-lo recorre-se ao método de suavização de Laplace (Laplace Estimator), que é uma técnica para suavizar dados categóricos. Este método, consiste em realizar uma correção de pequena amostra adicionando 1 a cada probabilidade calculada. Dessa forma, as probabilidades nunca serão zero (HUANG, 2017).

Todas as equações apresentadas anteriormente são aplicadas apenas a atributos discretos. Caso o conjunto de atributos seja contínuo, é aplicada uma função de densidade de probabilidade (FDP) para preservar os valores contínuos. Essa função irá calcular as regiões de probabilidades superiores e inferiores para os valores de uma variável aleatória, visto que só é possível avaliar a probabilidade de um atributo contínuo dentro de determinado intervalo. Nesse caso, deve-se assumir que os valores dos atributos tem uma distribuição normal. Logo, pode-se calcular a média e o desvio padrão de cada variável contínua usando a base de aprendizagem. Para estimar a probabilidade aplica-se a fórmula da FDP para a distribuição normal, sendo ela:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{9}$$

onde:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{10}$$

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu)^2 \tag{11}$$

O classificador Naive Bayes possui um treinamento rápido (varredura única), onde para cada nova instância são realizados vários cálculos de probabilidades "unidirecional", pois apenas a maior probabilidade é considerada relevante. Além disso, o classificador lida com dados contínuos, reais e discretos. Este método também não sensível a características irrelevantes. A principal desvantagem do Naive Bayes é o fato de assumir o pressuposto da independência dos atributos. De certa forma, por se tratar de um cenário irreal, a performance do classificador é afetada.

3.2 K-NEAREST NEIGHBORS (kNN)

Este método de funcionamento simples, mas efetivo em muitos casos de classificação, é um dos mais utilizados na área de aprendizagem de máquina (DINIZ, 2016). Ele parte do pressuposto que objetos relacionados ao mesmo conceito apresentam similaridades entre si, ou seja, o grau de disparidade entre dois vetores de atributos influencia na classificação do seu rótulo. O kNN, como o próprio nome sugere, é um algoritmo baseado na procura dos k vizinhos mais

próximos de uma determinada instância, ou seja, uma nova entrada (instância) será classificada (rotulada) baseada na sua proximidade com os seus vizinhos. Então, o algoritmo calcula a distância do elemento dado para cada elemento da base de treinamento e seleciona os k primeiros elementos mais próximos.

O cálculo da distância pode ser realizado utilizando qualquer métrica para calcular a distância entre dois pontos em um espaço euclidiano, como por exemplo as distâncias: Euclidiana, Manhattan, Euclidiana normalizada, Minkowski, etc. Logo, após realizar os cálculos de proximidade e formar um conjunto t de vizinhos mais próximos a determinada instância s, é necessário rotular este conjunto de atributos (instância) a partir de determinada regra de classificação. O método mais clássico para classificar um vetor de atributos utilizando o kNN, é considerar a votação por maioria entre os registros de dados da vizinhança, ou seja, o rótulo atribuído a s será baseado na classe de maior frequência do conjunto de vizinhos t (GUO, 2003). Dentre as métricas citadas anteriormente, a mais utilizada é a distância Euclidiana, descrita na equação (12).

$$D_E(p,q) = \sqrt{(p_1 - q_1)^2 + ... + (p_n + q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i + q_i)^2} \tag{12}$$

O algoritmo kNN é influenciado diretamente pelo valor de k ($k \geq 1$). Este, representa a quantidade de instâncias que serão selecionadas do conjunto dos vizinhos mais próximos. O valor mínimo que k pode assumir é 1. No entanto, para aplicar o kNN, necessita-se encontrar um valor apropriado para k, pois a performance do algoritmo é dependente desse valor. Existem diversas maneiras de escolher o melhor valor possível de k, mas uma delas é executar muitas vezes o algoritmo com diferentes valores de k e selecionar aquele com o melhor desempenho (GUO, 2003).

Na implementação do classificador kNN, é importante definir qual será a regra de classificação e a função que irá calcular a distância entre duas instância. Esta função, é responsável por mensurar a proximidade entre dois elementos e formar o conjunto dos vizinhos mais próximos. Já a regra de classificação tem o objetivo de tomar determinada decisão com os resultados da função de distância, ou seja, determinar qual a importância dos k vizinhos mais próximos selecionados. Como citado anteriormente, a frequência das classes do conjunto de vizinhança formados por k elementos de maior similaridade, é uma regra classificação clássica utilizada no trabalho de Gongde et al. Outra regra clássica citada no trabalho de Diniz et al. é considerar o peso pela distância, onde determinada instância será classificada de acordo com a soma dos pesos dos k vizinhos – o peso é inversamente proporcional as distâncias.

O classificador kNN é vantajoso por ser um método simples e de fácil implementação. Apesar da sua simplicidade, ele apresenta ótimos resultados dependendo de sua aplicação e seleção do seu parâmetro k, como também a escolha da função de classificação. Além disso, é um algoritmo bastante flexível, pois o usuário é livre para ajustar o seu funcionamento, seja através do parâmetro livre k ou por qual modelo matemática de distância (Euclidiana, Manhattan, etc)

adotar. A principal desvantagem do kNN é classificar um exemplo desconhecido, pois pode ser um processo computacionalmente complexo. Dado que cada novo elemento requer um cálculo de distância para cada instância da base de treinamento. Logo, para conjuntos de dados grande o tempo de processamento do algoritmo aumenta significativamente (LIMA, 2011).

3.3 ÁRVORE DE DECISÃO (AD)

De um modo geral em computação, árvores são estruturas de dados hierárquicas formadas por um conjunto de elementos que armazenam informações em suas "folhas", também denominadas de nós. Toda árvore possui um nó raiz, que possui o maior nível hierárquico e ligações para outros elementos, considerados filhos que por sua vez também possuem conexões com seus filhos. O nó que não possui filho é conhecido como nó folha ou terminal. Portanto, uma árvore de decisão nada mais é que uma árvore de dados que armazena regras em seus nós, e os nós folhas representam a decisão a ser tomada (CAMPOS, 2017).

Com isso, árvore de decisão é um modelo estatísticos não paramétrico (termo estatístico que significa, em termos gerais: não assumir nada sobre os dados de antemão) que utiliza um treinamento supervisionado para a classificação dos dados (CAMPOS, 2017). Sua estrutura é arranjada hierarquicamente e determinada por meio do aprendizado. Em outras palavras, em sua construção é utilizado um conjunto de treinamento formado por entradas e saídas. Estas últimas são as classes, que correspondem as folhas da árvore. O processo de classificação de um novo caso (instância) se inicia pelo nó raiz da árvore, e esta é percorrida, através das repostas aos testes dos nós internos, até que se chegue em um nó terminal, o qual indica a classe correspondente da nova instância. Percebe-se que o custo computacional da árvore no processo de classificação é proporcional a quantidade de nós internos, logo, há dependência do tempo de processamento de cada nó.

Na modelagem computacional de classificação geralmente emprega-se um dentre dois paradigmas alternativos, sendo eles: top-down e bottom-up. Na abordagem top-down a obtenção dos modelos de classificação é realizada a partir de informações fornecidas por especialistas. Já na abordagem bottom-up, obtêm-se o modelo classificador pela identificação de relacionamentos entre variáveis dependentes e independentes em bases de dados rotuladas. O classificador é induzido por mecanismos de generalização fundamentados em exemplos específicos (conjunto finito de objetos rotulados). Existem propostas também para dados não rotulados. Portanto, as árvores de decisão estão fundamentadas na abordagem bottom-up (VON ZUBEN; ATTUX, 2019).

O modelo supervisionado em questão, implementa a estratégia de dividir para conquistar, ou seja, a decomposição de um problema complexo em subproblemas mais simples e recursivamente esta técnica é aplicada para cada subproblema. Em outras palavras, criam-se subárvores a partir da raiz até chegar nas folhas, o que implica em uma divisão hierárquica em múltiplos subproblemas de decisão, os quais tendem a ser mais simples que o problema original.

As árvores de decisão apresenta uma tarefa de classificação bastante conhecida e aplicada

para diagnóstico médico, em que cada para cada paciente são definidos atributos contínuos ou categóricos ordinais (Ex: idade, altura, peso, temperatura do corpo, batimento cardíaco, pressão, etc.) e atributos categóricos não ordinais (Ex: sexo, cor da pele, local da dor, etc.). A tarefa do classificador é realizar um mapeamento dos atributos para um diagnóstico (Ex: saudável, pneumonia, Influenza A, etc.). Após a construção, esse classificador já pode ser usado para predizer o conjunto de teste, que contem classes completamente desconhecidas pelo algoritmo. Este classificador é bastante empregado em problemas de classificação por causa do conhecimento adquirido ser representado por meio de regras, permitindo que sejam expressas em linguagem natural, facilitando o entendimento por parte das pessoas (VON ZUBEN; ATTUX, 2019).

O TDIDT é um algoritmo bem conhecido e utilizado para indução de árvores de decisão. Esse algoritmo é responsável pela construção das regras de decisão e por definir quais as informações mais relevantes devem ser armazenadas por cada nó interno da árvore. O critério utilizado no processo de particionamento recursivo dos atributos para construção das árvores de decisão, é o da utilidade de cada atributo para classificação. O atributo que será escolhido para compor determinado nó é aquele que possui o maior ganho de informação. Observa-se que o algoritmo tratado é de busca gulosa que procura, sobre determinada conjunto de atributos, aqueles que "melhor"dividem o conjunto de exemplos em subconjuntos (VON ZUBEN; ATTUX, 2019).

Normalmente, quando trata-se de de árvores aplicadas para classificação, os critérios de partição, processo de escolha dos atributos de maior significância, mais conhecidos são baseados na entropia e no índice Gini.

A entropia é o cálculo do ganho de informação baseado em uma medida utilizada na teoria da informação. Ela caracteriza a impureza, desordem e incerteza de um determinado conjunto de dados, calculando a falta de homogeneidade dos dados de entrada em relação a sua classificação (OGURI, 2006). A entropia controla o particionamento da árvore de decisão, a partir dela é possível determinar qual informação determinado nó irá usar para teste, ou seja, define o dado armazenado em cada nó interno. A fórmula da entropia para menos de duas classe é descrita na equação (13).

$$Entropia(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \tag{13}$$

Onde, S é uma amostra dos exemplos de treinamento, $p_{\oplus}$ é a proporção de exemplos positivos em S e $p_{\ominus}$ é a proporção de exemplos negativos em S.

Já a equação (14) descreve a fórmula da entropia para mais de duas (n) classes.

$$Entropia(S) = \sum_{i=1}^{n} -p_i \log_2 p_i \tag{14}$$

Sendo, p_i é a proporção de dados em S que pertencem à classe i.

O ganho de informação (gain) mede o quanto de "informação"uma característica nos dá

sobre a classe. O gain para um atributo A de um conjunto de dados S nos dá a medida da diminuição da entropia esperada quando utilizamos o atributo A para fazer a partição do conjunto de dados. Um atributo com maior ganho de informação será testado/particionado primeiro (OGURI, 2006). A fórmula do ganho de informação é descrita por:

$$Gain(S,A) = Entropia(S) - \sum_{x \in P(A)} \frac{|S_x|}{|S|} Entropia(S_x) \tag{15}$$

onde, P(A) é um conjunto com os possíveis valores do atributo A.

Outra medida bastante conhecida, como citada anteriormente, é o Gini, que emprega um índice de dispersão estatística proposto por Conrado Gini em 1912. Este índice é muito utilizado em análises econômicas e sociais, por exemplo, para quantificar a distribuição de renda em um certo país (VON ZUBEN; ATTUX, 2019). Quando aplicados as árvores de decisão, o índice Gini tem a finalidade de medir a impureza de um atributo D, de uma partição de dados ou de um conjunto de tuplas de treinamento. A fórmula que melhor descreve o índice Gini é descrita por:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{16}$$

onde pi é a probabilidade da tupla em D pertencer à classe C_i. Se o índice for igual a zero, o nó é puro. Caso contrário, quanto mais próximo do valor 1, mais impuro é o nó.

A construção de uma árvore de decisão tem três objetivos, quais sejam: diminuir a entropia (a aleatoriedade da variável objetivo), ser consistente com o conjunto de dados e possuir o menor número de nós (OGURI, 2006). Portanto, pode-se inferir que uma das vantagens da árvore de decisão é a sua estrutura de fácil implementação, sendo possível estruturá-la a mão. Além disso, os seus modelos produzidos podem ser facilmente interpretados por humanos. Como o número de nós internos influencia diretamente no custo computacional, as árvores de decisão apresentam pouca robustez a dados de grande dimensão. Outra desvantagem é a sensibilidade a atributos de baixa significância, que, consequentemente, afeta a acurária do classificador.

3.4 SUPPORT VECTOR MACHINE (SVM)

A máquina de vetores de suporte é um algoritmo simples e bastante utilizado por especialistas em aprendizado de máquina, pois produz uma precisão significativa com menos poder de computação. O SVM pode ser utilizado em tarefas com objetivos de regressão, mas é amplamente empregado para classificação de dados (GANDHI, 2018).

O Support Vector Machine é um classificador tem como objetivo encontrar um hiperplano em um espaço n-dimensional que distintamente classifica os dados. Em outras palavras, ele busca encontrar o hiperplano que melhor se adapte aos dados, ou seja, o hiperplano de separação que maximize a margem dos dados de treinamento (KOWALCZYK, 2017). Em um problema com duas classes (bidimensional), por exemplo, podem existir n hiperplanos de separação dis-

tintos, porém o algoritmo do SVM encontra o hiperplano que apresenta a margem máxima, ou seja, a distância máxima entre pontos de dados de ambas as classes e o hiperplano. Maximizar a distância da margem fornece algum reforço para que os pontos de dados futuros possam ser classificados com mais confiança (GANDHI, 2018). Logo, o algoritmo SVM funciona dessa forma, encontra o hiperplano capaz de generalizar melhor os dados ocultos, desconhecidos pelo modelo.

Os hiperplanos indicam os limites de decisão que ajudam no processo de classificação dos dados, ou seja, estabelece uma fronteira entre os rótulos de determinado conjunto de dados. Os pontos de dados localizados em lados opostos ao hiperplano são classificados com classes distintas. Além disso, a dimensão do hiperplano depende do número de classes. Caso a dimensão dos rótulos seja unidimensional (apenas um rótulo) o hiperplano de separação é chamado de ponto. Se a dimensão for bidimensional representa-se o hiperplano com uma linha. No caso da dimensão ser tridimensional, o hiperplano é representado por um plano. Já em dimensões maiores que a tridimensional separa-se os dados por meio de um hiperplano (GANDHI, 2018).

O SVM pode ser aplicado a conjuntos de treinamento linearmente separáveis ou não. Um determinado conjunto S é dito linearmente separável se é possível separar os padrões das classes contido no mesmo por pelo menos um hiperplano (LORENA; CARVALHO, 2003). Logo, classificadores que separam os dados por meio de um hiperplano são denominados lineares, podendo ser definidos pela seguinte equação:

$$w \cdot x + b = 0 \tag{17}$$

onde, $w \cdot x$ é o produto escalar entre os vetores w e x, em que w é o vetor normal ao hiperplano, b é um termo "compensador". O par (w, b) é determinado durante o treinamento do classificador. Esta equação divide o espaço de entradas em duas regiões: $w \cdot x + b > 0$ e $w \cdot x + b < 0$, levando as desigualdades (18) (LORENA; CARVALHO, 2003).

$$y_i = \begin{cases} 1, & \text{se} \quad w \cdot x + b > 0 \\ -1, & \text{se} \quad w \cdot x + b < 0 \end{cases} \tag{18}$$

Pode-se aplicar uma função sinal $g(x) = sgn(f(x)) = sgn(w \cdot x + b)$, sendo $f(x)$ a função dada pela equação (17) do hiperplano, levando a classificação $+1$ se $f(x) > 0$ e -1 se $f(x) < 0$. Logo, um conjunto de treinamento é linearmente separável se é possível determinar pelo menos um par (w, b) tal que a função de sinal consiga classificar corretamente todos os exemplos contidos neste grupo (LORENA; CARVALHO, 2003).

Considerando que o conjunto de treinamento é linearmente separável sem erros (ruídos ou valores discrepantes). Com isso, os classificadores que separam tal conjunto assumem margem positiva a partir das desigualdades (19). Segundo este sistema de inequações, não há pontos entre o ponto de dados mais próximo e o hiperplano de separação, ou seja, supõe-se que a margem ρ é sempre maior que a distância entre os hiperplanos $w \cdot x + b = 0$ e $|w \cdot x + b = 1|$.

Partindo dessa suposição, as SVMs lineares obtidas são usualmente denominadas de SVMs com margens rígidas (Hard-Margin). Elas definem fronteiras lineares a partir de dados linearmente separáveis (LORENA; CARVALHO, 2003).

$$\begin{cases} w \cdot x_i + b \geq 1, \text{ se } y_i = 1 \\ w \cdot x_i + b \leq -1, \text{ se } y_i = -1 \\ i = 1, ..., n \end{cases} \tag{19}$$

Supondo que os valores arbitrários x_1 e x_2 são pontos de diferentes classes e que x_1 intercepta a reta perpendicular a x_2 (são os pontos mais próximos ao hiperplano de separação). Logo, tem-se que:

$$\begin{cases} w \cdot x_1 + b = -1 \\ w \cdot x_2 + b = 1 \end{cases} \tag{20}$$

A partir da resolução do sistema (20), obtém-se a seguinte equação:

$$w \cdot (x_2 - x_1) = 2 \tag{21}$$

Como w e $x_2 - x_1$ são ortogonais ao hiperplano separador, esses vetores são paralelos entre si. Deduzindo a equação (21), tem-se:

$$|w \cdot (x_2 - x_1)| = ||w|| \times ||x_2 - x_1|| \tag{22}$$

Substituindo (21) em (22), tem-se:

$$||x_2 - x_1|| = \frac{2}{||w||} \tag{23}$$

Como x_2 e x_1 são os valores que limitam a margem, a norma $||x_2 - x_1||$ mede a distância entre os hiperplanos $w \cdot x + b = -1$ e $w \cdot x + b = 1$. Logo, pela equação (23) pode-se afirmar que a distância entre esses hiperplanos é dada por $\frac{2}{||w||}$. Essa distância é denominada de margem, representada por ρ (LORENA; CARVALHO, 2003).

A margem (ρ) de determinado hiperplano é representada pelo dobro da distância entre o hiperplano e o ponto de dados mais próximo a ele (delimitadores da margem). Como o objetivo do SVM é encontrar a margem máxima, percebe-se que quanto menor a distância entre o hiperplano e um ponto de dados, sua margem será minimizada. Portanto, a medida que a distância entre o hiperplano e os delimitadores da margem aumenta, a margem também cresce (LORENA; CARVALHO, 2003). Isso significa, como dito anteriormente, que o hiperplano ideal será aquele com a maior margem. Para calcular a margem utiliza-se a seguinte equação:

$$\rho = \frac{2}{||w||} \tag{24}$$

onde, $||w||$ é a norma do vetor normal ao hiperplano. Portanto, o hiperplano ótimo é definido para os valores de w e b que satisfazem as desigualdades (19) e para os quais a norma $||w||$ é mínima. Com isso, o objetivo é encontrar o hiperplano que maximize ρ ou minimize a norma euclidiana de w ($||w||$) (LORENA; CARVALHO, 2003). Logo, é deduzido o seguinte problema:

$$\rho = max \frac{2}{||w||} \tag{25}$$

ou

$$\rho = min \frac{1}{2}||w||^2 \tag{26}$$

$$\text{Sob as restrições:} \quad y_i(w \cdot x_i + b) - 1 \geq 0, \forall_i = 1, ..., n$$

Como o referido problema de otimização é restrito, apresentando restrições em relação à w. Além de tratar-se de um problema convexo, onde apresenta apenas um mínimo global (quando viável), fato que torna o algoritmo SVM atrativo. Portanto, pode-se utilizar o método dos multiplicadores de Lagrange para encontrar os pontos ótimos (LORENA; CARVALHO, 2003). A função Lagrangiana, definida em termos de w e b, é apresentada na equação (27) (LORENA; CARVALHO, 2003).

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i(w \cdot x_i + b) - 1) \tag{27}$$

Onde, os α_i são denominados multiplicadores de Lagrange. O problema passa a ser então a minimização da equação (238 em relação a w e b e a maximização dos α_i. Os pontos ótimos da equação (27) podem ser calculados a partir da resolução das igualdades apresentadas em (28) e (29) (LORENA; CARVALHO, 2003).

$$\frac{\partial L}{\partial b} = 0 \tag{28}$$

$$\frac{\partial L}{\partial w} = 0 \tag{29}$$

Resolvendo as equações (28) e (29), obtém-se o seguinte resultado representado pelas equações (30) e (31).

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{30}$$

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{31}$$

Substituindo (30) e (31) em (27), obtém-se o seguinte problema de otimização (denominado dual):

$$\textbf{Maximizar:} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{32}$$

$$\textbf{Sujeito a:} \begin{cases} \alpha_i \geq 0, i = 1, ..., n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

Uma vez calculado o valor ótimo α^*, o valor ótimo do vetor normal ao hiperplano w^* é encontrado a partir da equação (31). Sendo determinada a direção do hiperplano ótimo (ou seja, w^*), o valor ótimo b^* pode ser calculado pela seguinte equação:

$$\begin{aligned} b^* &= -\frac{1}{2} \left[\max_{\{i|y_i=-1\}} (w^* \cdot x_i) + \min_{\{i|y_i=+1\}} (w^* \cdot x_i) \right] \\ &= -\frac{1}{2} \left[\max_{\{i|y_i=-1\}} \left(\sum_{i=1}^{n} y_j \alpha_j^* x_i \cdot x_j \right) + \min_{\{i|y_i=+1\}} \left(\sum_{i=1}^{n} y_j \alpha_j^* x_i \cdot x_j \right) \right] \end{aligned} \tag{33}$$

Segundo as fórmulas para determinação do hiperplano ótimo, tenta-se encontrar os valores de α^* a fim de determinar (w^*,b^*). Os valores de α_i^* assumem valores positivos para exemplos do treinamento que estão a uma distância do hiperplano ótimo exatamente igual à margem. Para outros padrões, o valor de α_i^* é nulo. Os dados que possuem $\alpha_i^* > 0$ são denominados vetores de suporte (Support Vectors - SV) e podem ser considerados os dados mais informativos do conjunto de treinamento, pois somente eles participam na determinação da equação do hiperplano separador (LORENA; CARVALHO, 2003).

Como resultado final, tem-se o classificador $g(x)$ apresentado na equação (34), em que *sgn* representa a função de sinal, w^* é fornecido pela equação (31), b^* pela equação (33) e *SV* é o conjunto dos vetores de suporte.

$$g(x) = sgn(f(x)) = sgn\left(\sum_{x_i \varepsilon SV} \alpha_i^* y_i x_i \cdot x + b^* \right) = \begin{cases} +1, \text{ se } \sum\limits_{x_1 \in SV} \alpha_i^* y_i x_i \cdot x + b^* > 0 \\ -1, \text{ se } \sum\limits_{x_1 \in SV} \alpha_i^* y_i x_i \cdot x + b^* < 0 \end{cases} \tag{34}$$

A função linear expressa na equação (34) representa o hiperplano que separa os dados com maior margem, considerado aquele com melhor capacidade de generalização. Essa característica difere as SVMs lineares de margens rígidas das Redes Neurais Perceptron, em que o hiperplano obtido na separação dos dados pode não corresponder ao de maior margem de separação (LORENA; CARVALHO, 2003).

Em situações reais, é difícil encontrar um conjunto de dados que sejam linearmente separáveis e livres de ruídos. Isso vai depender de diversos fatores, como por exemplo, a pre-

sença de ruídos, que influencia diretamente na determinação do hiperplano ótimo separador. É possível estender o modelo das SVMs lineares de margens rígidas para tratarem de problemas mais gerais. Portanto, permite-se que alguns dados possam está entre a região delimitadora conhecida como margem, ou seja, esses dados não respeitariam totalmente a restrição $y_i(w \cdot x_i + b) - 1 \geq 0$, para $i = 1, ..., n$. Para isso, são introduzidas variáveis de folgas ξ_i, para todo $i = 1, ..., n$ (LORENA; CARVALHO, 2003). Logo, essas variáveis suavizam as restrições impostas ao problema de otimização, que se tornam:

$$y_i(w \cdot x_i + b) - 1 \geq -\xi_i,\ \xi_i \geq 0,\ \forall_i = 1, ..., n \tag{35}$$

A aplicação dessa restrição suaviza as margens do classificador linear, permitindo a ocorrência de erros de classificação. Por esse motivo, as SVMs obtidas neste caso também podem ser referenciadas como SVMs com margens suaves. Um erro identificado no conjunto de treinamento é indicado por um valor ξ_i maior que 1. Logo, o somatório de todos os ξ_i representa um limite no número de erros de treinamento. Considerando o termo ξ_i, a equação (26) reformulada como (LORENA; CARVALHO, 2003):

$$\rho = min \frac{1}{2}||w||^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \tag{36}$$

onde, a constante C é um termo de regularização que impõe um peso à minimização dos erros no conjunto de treinamento em relação à minimização da complexidade do modelo. A presença do termo $\sum_{i=1}^{n} \xi_i$ o problema de otimização também pode ser vista como uma minimização de erros marginais, pois um valor de ξ_i ε (0, 1] indica um dado entre a margem.

Como, o problema de otimização gerado é quadrático, considerando as novas restrições da equação (35). Portanto, também aplica-se a função Lagrangiana para determinar os pontos ótimos, além de tornar suas derivadas parciais nulas. Logo, tem-se o seguinte problema de otimização (problema dual):

$$\underset{\alpha}{Maximizar} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{37}$$

$$\textbf{Sujeito a:} \begin{cases} 0 \geq \alpha_i \geq C, \forall_i = 1, ..., n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

Observa-se que essa formulação é semelhante aos SVMs de margens rígidas, a não ser pela restrição nos α_i, que agora são limitados pelo valor de C. O vetor w^* continua sendo representado pela equação (31). Já os valores de ξ_i^* são calculados pela seguinte equação:

$$\xi_i^* = max\left\{0, 1 - y_i \sum_{j=1}^{n} y_j \alpha_j^* x_j \cdot x_i + b^*\right\} \tag{38}$$

Como nas SVMs de margens rígidas, os pontos x_i para os quais $\alpha_i^* > 0$ são denominados

vetores de suporte (SVs), sendo os dados que participam da formação do hiperplano separador. Porém, no caso do modelo de margens suaves pode-se distinguir tipos distintos de SVs. Os vetores de suporte para os quais $\alpha_i^* = C$ podem representar três casos: erros, se $\xi_i^* = 0$; pontos corretamente classificados, porém entre as margens, se $0 < \xi_i^* \leq 1$; ou pontos sobre as margens, se $\xi_i^* = 0$. Por fim, o resultado final de classificação é semelhante ao das SVMs de margens rígidas, porém as variáveis ótimas α_i^* são determinadas pela solução e restrições da equação (37).

As SVMs lineares são eficazes na classificação de conjuntos de dados linearmente separáveis ou que possuam uma distribuição aproximadamente linear, sendo que a versão de margens suaves tolera a presença de alguns ruídos (LORENA; CARVALHO, 2003). Portanto, quando os padrões não são lineares o hiperplano de separação não representará um resultado satisfatório na divisão do conjunto de dados de treinamento.

Para lidar com problemas não lineares as SVMs mapeam não linearmente o vetor de entrada para um espaço de característica (feature space) de mais alta dimensionalidade. Pois, se forma mais adequada de separação das classes de determinando conjunto de treinamento fosse uma curva o SVM linear até conseguiria achar um hiperplano, porém os erros residuais seriam altos e o hiperplano não se adequaria totalmente aos dados. Logo, para realizar esse mapeamento são definidas funções reais Φ_1,...,Φ_M no domínio de entrada. Por meio da utilização destas funções, mapeia-se o conjunto de treinamento para o espaço de características, da maneira apresentada a seguir (LORENA; CARVALHO, 2003):

$$\begin{aligned} x_i (i = 1, ..., n) &\longmapsto \Phi(x_i) = (\Phi_1(x_i), \Phi_2(x_i), ..., \Phi_M(x_1)) \\ &\Rightarrow \Phi(S) = \{(\Phi(x_1), y_1), (\Phi(x_2), y_2), ..., (\Phi(x_n), y_n)\} \end{aligned} \tag{39}$$

onde, S é o conjunto de treinamento e $\Phi(S)$ é o espaço de característica. Uma característica singular deste espaço é que a escolha de uma função Φ apropriada torna o conjunto de treinamento linearmente separável. Logo, as SVMs lineares apresentadas anteriormente podem utilizar o conjunto de treinamento mapeados (LORENA; CARVALHO, 2003).

Considere um exemplo em que um limite circular é mais adequado para separação das classes, ou seja, um conjunto de dados não linearmente separável. Logo, utilizando a transformação deste conjunto com uma função Φ adequada pode-se obter um mapeamento de $\Re^2$ para $\Re^3$, em que o conjunto de dados não linear em $\Re^2$ torna-se linearmente separável em $\Re^3$. Com isso, é possível encontrar um hiperplano capaz de separar esses dados e ter uma boa generalização. Para realizar o mapeamento, aplica-se Φ aos exemplos presentes no problema de otimização da equação (37) e sob as mesmas restrições, conforme ilustrado a seguir (LORENA; CARVALHO, 2003):

$$\underset{\alpha}{Maximizar} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) \tag{40}$$

De forma semelhante as equações apresentadas anteriormente, o classificador extraído se torna (LORENA; CARVALHO, 2003):

$$g(x) = sgn(f(x)) = sgn\left(\sum_{x_i \varepsilon SV} \alpha_i^* y_i \Phi(x_i) \cdot \Phi(x) + b^*\right) = \begin{cases} +1, \text{ se } \sum\limits_{x_1 \in SV} \alpha_i^* y_i \Phi(x_i) \cdot \Phi(x) + b^* > 0 \\ -1, \text{ se } \sum\limits_{x_1 \in SV} \alpha_i^* y_i \Phi(x_i) \cdot \Phi(x) + b^* < 0 \end{cases} \tag{41}$$

Em que b^* é adaptado também aplicando o mapeamento aos dados:

$$b^* = -\frac{1}{2}\left[\max_{\{i|y_i=-1\}}\left(\sum_{i=1}^{n} y_j \alpha_j^* \Phi(x_i) \cdot \Phi(x)\right) + \min_{\{i|y_i=+1\}}\left(\sum_{i=1}^{n} y_j \alpha_j^* \Phi(x_i) \cdot \Phi(x)\right)\right] \tag{42}$$

A partir das equações listadas na demonstração (39), percebe-se que a única informação necessária sobre o mapeamento Φ é uma definição de como o produto interno $\Phi(x_i) \cdot \Phi(x_j)$ pode ser realizado, para quaisquer x_i e x_j pertencentes ao espaço de entradas. Isto é obtido com a introdução do conceito de Kernels.

A função de Kernel, citada anteriormente, simplesmente computa o produto escalar no espaço de características a partir de duas variáveis x_i e x_j de entrada, como descrito em (LORENA; CARVALHO, 2003):

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{43}$$

Por fim, não precisamos mapear explicitamente os dados no espaço de alta dimensão para resolver o problema de otimização (para treinamento). Logo, reformulando a equação (41) tem-se o seguinte classificador:

$$g(x) = sgn(f(x)) = sgn\left(\sum_{x_i \varepsilon SV} \alpha_i^* y_i K(x_i, x) + b^*\right) = \begin{cases} +1, \text{ se } \sum\limits_{x_1 \in SV} \alpha_i^* y_i K(x_i, x) + b^* > 0 \\ -1, \text{ se } \sum\limits_{x_1 \in SV} \alpha_i^* y_i K(x_i, x) + b^* < 0 \end{cases} \tag{44}$$

Sendo assim, as SVMs embasadas em teorias estatística expressa o aprendizado em modelo matemático aproveitando a rica teoria do problema de otimização. O SVM utiliza o truque do Kernel, mapeando um conjunto de dados de treinamento para uma alta dimensão, a fim de tornar o problema não linear resolvível com equações do modelo de SVM linear. Outra característica atrativa é a convexidade do problema de otimização formulado em seu treinamento, que implica na existência de um único mínimo global. Entre as principais limitações das SVMs encontram-se a sensibilidade aos parâmetros e a dificuldade de interpretação do modelo gerado por essa técnica. Uma outra desvantagem está relacionado ao tempo de treinamento que pode ser bem longo dependendo do número de exemplos e dimensionalidade dos dados. Outro fator que

influencia no processo de classificação é a definição de um bom Kernel, que consiga mapear os dados em um espaço de alta dimensão, cujo hiperplano encontrado seja o melhor possível.

3.5 REDES NEURAIS ARTIFICIAIS (RNA's)

Na neurociência, as Redes Neurais Biológicas (RNB) consistem em um conjunto finito de neurônios conectados uns aos outros por meio de sinapses. Estas, são responsáveis por criar um canal de transmissão de estímulos através de diferentes concentrações de Na_+ (Sódio) e K_+ (Potássio), e o resultado disto pode ser estendido por todo o corpo humano. Basicamente, todas os movimentos e pensamentos são resultados de impulsos eletromagnéticos coordenados pelos neurônios. Além disso, esta grande rede proporciona uma excelente capacidade de processamento e armazenamento (TATIBANA; KAETSU, 2004).

O sistema nervoso é formado por um conjunto complexo de neurônios, unidade básica das redes neurais. A complexidade da conectividade entre os neurônios é responsável pelas características atribuídas à inteligência. As principais componentes dos neurônios são: os dendritos, responsáveis por receber os estímulos transmitidos pelos outros neurônios; o corpo de neurônio, que é responsável por coletar e combinar informações vindas de outros neurônios; e o axônio, responsável por transmitir os estímulos para outras células (TATIBANA; KAETSU, 2004).

Os primeiros registros relacionados a neurocomputação datam de 1943, em artigos de McCulloch e Pitts, em que sugeriram um modelo de um máquina inspirada no cérebro humano. O modelo sugerido é formado por um vetor de entradas, onde as sinapses representam os pesos numéricos, por uma função soma, que é responsável por realizar a soma ponderada das entradas e submeter a função de ativação (ou transferência), que determina se a soma possui um valor numérico maior que o limiar do neurônio. Se sim o neurônio é ativado, caso contrário é desativado (TATIBANA; KAETSU, 2004). O funcionamento é muito simples, o neurônio apenas responde se a soma recebida é maior que o seu limiar.

Os modelos neurais propostos pela neurocomputação, procuram aproximar o processamento dos computadores ao cérebro. As redes neurais possuem um grau de interconexão similar a estrutura dos cérebro (TATIBANA; KAETSU, 2004). Com isso, as redes neurais consistem em um método capaz de solucionar problemas de modo similar ao cérebro humano, ou seja, o encapsulamento das RNB's em um modelo matemático de inteligência artificial. Em outras palavras, o intuito é assimilar determinado problema da mesma forma que faz uma rede neural biológica: aprendendo, errando e fazendo descobertas. A fórmula que melhor descreve o modelo matemático proposto por McCulloch e Pitts é decrita na equação a seguir.

$$y = g(\sum_{i=1}^{n} w_i x_i) \tag{45}$$

Onde, x_i representa o sinal de entrada (i=1,2,...n), w_i os pesos (i=1,2,...,n), $g(u)$ função de ativação e y sinal de saída.

O modelo de McCulloch e Pitts é formado por neurônios que possuem atividades binária,

ou seja, a cada instante o neurônio, ou está disparando (atividade = 1), ou não está disparando (atividade = 0). O modelo neural é constituído por linhas direcionadas (ligações), sem pesos, ligando os neurônios. Essas linhas podem ser excitatórias ou inibitórias (positivas ou negativas). Outra característica é que cada neurônio tem um limiar fixo L, de maneira que uma vez ultrapassado esse limite o neurônio dispara, ou seja, transmite informações para outros neurônios. A chegada de uma única sinapse inibitória num dado instante evita absolutamente o disparo do neurônio, independentemente do número de sinapses excitatórias que estejam chegando conjuntamente com a sinapse inibitória. Um detalhe que torna esse modelo uma reprodução do cérebro humano, é que a transmissão de informações de um neurônio da rede para outro leva uma unidade de tempo. Isso é uma tentativa de reproduzir os atrasos sinápticos existentes na RNB (TOGURO, 2010).

Os pesos (sinapses) das ligações entre os neurônios são responsáveis por todo o aprendizado da rede neural, pois a partir de variações e correções nos valores dos pesos a RNA aproxima o valor predito do valor real. Em outras palavras, os pesos são determinantes na ativação do neurônio. Por exemplo, se um classificador de imagens obteve um resultado "7"para uma imagem "8", pode-se manualmente alterar os pesos para que a saída seja "8"atingindo um bom nível de classificação. Porém, essa atualização indevida nos pesos pode afetar o comportamento da rede neural de forma que outras imagens sejam classificadas incorretamente. Portanto, há uma maneira inteligente de resolver esse problema, que é introduzindo um componente matemático em nosso neurônio artificial, chamado função de ativação.

As funções de ativação são essenciais para dar capacidade representativa às redes neurais artificiais, introduzindo um componente de não linearidade. Por outro lado, com esse poder a mais surgem algumas dificuldades. Ao introduzir uma ativação não linear, a superfície de custo da rede neural deixa de ser convexa, tornando complexo o processo de otimização (FACURE, 2017).

Essas funções de ativação permitem que pequenas mudanças nos pesos causem apenas uma pequena alteração na saída. Esse é o fato crucial que permitirá que uma rede de neurônios artificiais aprenda. Essas funções decidem se um neurônio deve ser ativado ou não. Ou seja, se a informação que o neurônio está recebendo é relevante para a informação fornecida ou deve ser ignorada. Existem diversos tipos de função de ativação, dentre elas tem-se as mais populares funções: etapa binária, linear, sigmoide e tangente hiperbólica (Tanh) (FACURE, 2017).

A função de etapa binária (em inglês, Binary Step Function), como o próprio nome indica, é baseada em um limiar (threshold), ou seja, se o neurônio deve ou não ser ativado, assumindo valor 1 e 0 para ativação e não ativação, respectivamente. Se o valor Y (representa a operação matemática realizada sobre os valores e pesos das sinapses) estiver acima de um valor de limite determinado, ative o neurônio senão deixa desativado (FACURE, 2017). Essa regra seria representada da seguinte forma:

$$\begin{aligned} f(x) &= 1,\ x \geq 0 \\ f(x) &= 0,\ x < 0 \end{aligned} \tag{46}$$

Essa função é destinada para classificadores binários, onde simplesmente precisa-se da tomada de decisão positiva (sim) ou negativa (não) para uma única classe. A função é mais teórica do que prática, pois, na maioria dos casos, classifica-se os dados em várias classes do que apenas uma única classe. A função de etapa não seria capaz de fazer isso. Além disso, o gradiente da função de etapa é zero. Isso faz com que a função de etapa não seja tão útil durante o backpropagation quando os gradientes das funções de ativação são enviados para cálculos de erro para melhorar e otimizar os resultados. O gradiente da função de etapa reduz tudo para zero e a melhoria dos modelos realmente não acontece (ACADEMY, 2019).

Visto o problema com a função etapa, o gradiente sendo zero, e a impossibilidade de atualizar o gradiente durante a backpropagation, pode-se tentar usar uma função de ativação linear para corrigir esse problema. Pode-se definir a função como:

$$f(x) = ax \tag{47}$$

A derivada de uma função linear é constante, isto é, não depende do valor de entrada x. Sua derivada é expressa po $f'(x) = a$, sendo a uma constante. Logo, durante a backpropagation o gradiente sempre seria o mesmo e não mais zero. E este é um grande problema, não estamos realmente melhorando o erro, já que o gradiente é praticamente o mesmo. Agora, se cada camada tiver uma transformação linear, não importa quantas camadas nós tenhamos, a saída final não é senão uma transformação linear da entrada. Portanto, a função linear pode ser ideal para tarefas simples, onde a interpretabilidade é altamente desejada (ACADEMY, 2019).

A função sigmoide ou logística é uma das mais utilizadas e possui um intervalo de variação entre 0 e 1. Essa função e sua derivada são dadas por:

$$\begin{aligned} \sigma(x) &= \frac{e^{px}}{1+e^{px}} = \frac{1}{1+e^{-px}} \\ \sigma'(x) &= p\sigma(x)(1-\sigma(x)) \end{aligned} \tag{48}$$

Esta é uma função suave e é continuamente diferenciável, além de ser não linear. Isto significa essencialmente que quando eu tenho vários neurônios com função sigmoide como função de ativação – a saída também não é linear. Essa função apresentam maior similaridade com o funcionamento sistemático biológico. Como neurônios biológicos funcionam de foma binária (ativando ou não ativando), a função sigmoide é uma boa forma de modelar esse comportamento, já que assume valores apenas entre 0 (não ativação) e 1 (ativação). Porém, com a sigmoide tem-se problemas quando os gradientes se tornam muito pequenos. Isso significa que

o gradiente está se aproximando de zero e a rede não está realmente aprendendo (ACADEMY, 2019).

Com base da função sigmoide e sua derivada, observa-se que irá existir pontos de saturação, mais especificamente para valores acima de 5 e abaixo de -5. Com essas derivadas tendendo a zero, a propagação do gradiente desvanece nessas regiões, causando dificuldades no treinamento. Outro fator que causa estas dificuldades é o fato da derivada sempre ser menor que 1, isso é problemático, fazendo com que desvaneça o produto dado pela regra da cadeia na propagação dos gradientes (ACADEMY, 2019).

A função tanh possui semelhança com a função sigmoide, porém a variação é de -1 a 1. Na verdade, é apenas um escalonamento da sigmoide. A tanh e sua derivada são dadas por:

$$\begin{aligned} tanh(x) &= 2\sigma(2x) - 1 \\ tanh'(x) &= 1 - tanh^2(x) \end{aligned} \tag{49}$$

A tanH se aproxima mais da identidade, sendo assim uma alternativa mais atraente do que a sigmoide para servir de ativação às camadas ocultas das RNAs. Nessa função existem saturações, mas o valor da derivada é maior, chagando ao máximo de 1 quando x = 0. Por esse motivo recomenda-se utilizar a tanh no lugar da logística, pois minimizam as dificuldades no treinamento em relação a esta última.

Essa breve introdução a respeito das funções de ativação mais comuns, é importante deixar claro o conceito de backpropagation e gradiente, mas antes, precisa-se falar sobre a arquitetura mais simples de uma rede neural artificial, o Perceptron. Este modelo, surgiu nas décadas de 1950 e 1960 desenvolvido pelo cientista Frank Rosenblatt, inspirado em trabalhos anteriores de Warren McCulloch e Walter Pitts. A sua abordagem é simplificada e permite uma compreensão clara de como funciona uma rede neural em termos matemáticos.

Um Perceptron é um modelo matemático que recebe n entradas, cada uma com seu respectivo peso, e produz uma única saída binária, que é determinada pela soma ponderada das entradas e seus pesos. Se essa soma ultrapassar um determinado valor limite, chamado de limiar (threshold), a saída será 1, caso contrário será 0. Geralmente, a soma ponderada é submetida a função sigmoide, tangente hiperbólica ou linear simples. Esse sistema de decisão do Perceptron é dado por:

$$sada = \begin{cases} 0, \text{se } \sum_j w_j x_j \geq threshold \\ 1, \text{se } \sum_j w_j x_j > threshold \end{cases} \tag{50}$$

O modelo matemático descrito acima, conhecido como Perceptron, tem a capacidade de aprender conceitos a partir de mudanças realizadas em seus pesos e limiar. O processo de aprendizado do Perceptron é simples, consiste apenas em verificar se a resposta para um dado vetor de um grupo de treinamento está correta ou não. Se a saída estiver correta, nenhuma

mudança é feita. Caso contrário, os pesos e as inclinações são atualizados usando as regras de aprendizado do perceptron. Os ajustes dos pesos são realizados a partir da seguinte equação:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i} \tag{51}$$

Onde, $w_i(t+1)$ é o novo peso, que é ajustado levando em conta: o peso anterior e o erro. O erro é calculado pela diferença entre o valor esperado (d_j) e o valor encontrado (y_j), multiplicado pela entrada associada ao seu respectivo peso e α, que é um valor chamado de taxa de aprendizado. Esta taxa, geralmente com um valor muito pequeno (como 0,000001) ajuda o perceptron a não "fugir"muito da solução ideal. Essa função de atualização é aplicada a cada par (x,t) do conjunto de treinamento.

Minsky e Papert, em 1969, introduziram um teorema chamada de "Teorema da Convergência", onde afirma que o algoritmo de aprendizado do Perceptron converge dentro de um número finito de passos para um vetor de pesos que classifica corretamente todo o conjunto de treinamento, considerando que este conjunto seja linearmente separável.

Um pouco depois do Perceptron ser introduzido no conceito de redes neurais artificais, surgiu a rede Adaline (Adaptative Linear Element) proposto por Widrow e Hoff, no ano de 1960. Esta rede possui a mesma estrutura do Perceptron, porém com as unidades de saída tendo funções de transferência lineares e com uma nova regra de aprendizado supervisionado, que ficou conhecida como "Regra de Widrow-Hoff"(ou "Regra Delta"). Assim como o Perceptron, a Adaline é uma rede neural tendo uma camada de entrada com N unidades e uma camada de saída com apenas uma unidade. Na Adaline, a atividade do neurônio de saída não é uma variável binária como no Perceptron, mas uma função linear do seu nível de ativação (ROQUE, 2016).

Quando um padrão é inicialmente apresentado à rede, ela produz uma saída. Após medir a distância entre a resposta atual e a desejada, são realizados os ajustes apropriados nos pesos das conexões de modo a reduzir esta distância. Este procedimento é conhecido como Regra Delta. Isso nos dá a seguinte equação de custo:

$$E(w) = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2 \tag{52}$$

onde, D é o conjunto de treinamento que se permanece fixo durante o treino, t_d é a saída desejada para o exemplo d e o_d é saída da função de ativação sem limiar (unthresholded) para o exemplo d. Caracteriza-se o E como uma função dos pesos.

O maior desafio é encontrar um vetor w ótimo que melhor se ajuste ao conjunto de treinamento. Para isso utiliza-se a técnica do gradiente descendente. Este, possui um funcionamento simples e baseia-se na ideia de encontrar o vetor de pesos que minimiza a função de custo E. Começando a partir de um vetor arbitrário w, calcula-se o gradiente da função de custo com respeito a cada peso. Sabe-se que o gradiente de uma determinada função define a direção de máximo crescimento desta, ou seja, calculando o gradiente encontra-se o valor que maximiza a

função de custo (ROQUE, 2016). Logo, deseja-se subtrair este gradiente dos pesos e para isso deve-se alterar os pesos na direção que produz a descida mais acentuada ao longo da superfície do erro. Portanto, o critério de parada desse ajuste e consequente minimização da função custo, é até que um mínimo global seja encontrado.

O gradiente da função de custo (E) é um vetor de derivadas parciais que especifica a direção de crescimento mais acentuado de E, sendo representado por: $\Delta E(w) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_n}]$. A regra do gradiente descendente tem como objetiva minimizar a função de custo a partir de modificações no vetor de peso proporcional ao negativo do gradiente, logo, tem-se a seguinte equação:

$$\Delta w = -\eta \Delta E(w) \tag{53}$$

onde, η é uma constante positva, chamada de taxa de aprendizado. O sinal negativo força o deslocamento do vetor de peso na direção que E decresce.

É possível observar que esta técnica do gradiente descendente, na busca pelo melhor vetor de ajuste, a convergência para um mínimo local pode depender de várias iterações, dependendo do problema envolvido. Outro dificuldade enfrentada é se existir vários mínimos locais na superfície de erro, pois não haverá garantia do algoritmo encontrar o mínimo.

Comparando a regra do Perceptron com a regra delta observa-se alguns pontos importantes em relação a atualização dos pesos. A regra do Perceptron é baseada no erro da saída após a aplicação do limiar. Enquanto, que na regra delta não há aplicação do limiar. No princípio de convergência, a regra Perceptron converge para um número finito de passos para encontrar um vetor de pesos que classifica corretamente o conjunto de treinamento, considerando que este seja linearmente separável. Já a regra Delta converge apenas assintoticamente para um vetor de pesos com um erro mínimo (técnica do gradiente descendente), com um número ilimitado de iterações. Neste técnica o conjunto de treinamento independe de ser linearmente separável.

Existem também as redes neurais perceptron multicamadas (em inglês, Multilayer Perceptron - MPL), que surgiu com o objetivo de superar determinadas limitações das redes de camada única, como é o caso da regressão linear, por exemplo. Essa arquitetura de rede possui duas ou mais camadas de neurônios para processamento, muitas vezes possuem as chamadas camadas ocultas (intermediárias), camadas que não são nem entrada e nem saída (IYODA, 2000).

O algoritmo backpropagation (em português, retropropagação) é uma técnica de aprendizado supervisionado mais utilizada no treinamento de redes neurais multicamadas com uma ou mais camadas escondidas. Basicamente, o funcionamento algoritmo de retropropagação consiste no processamento direto, onde uma entrada é aplicada à rede neural e seu efeito é propagado pela rede, camada a camada. Porém, a grande vantagem em utilizar esse algoritmo é justamente o princípio do processamento reverso, onde um sinal de erro calculado na saída da rede é propagado no sentido reverso, camada a camada, e ao final os pesos são ajustados de acordo com uma regra de correção de erro (IYODA, 2000).

3.6 ENSEMBLES

Ensemble (ou comitê, também conhecido como máquina de classificadores) é um paradigma de aprendizado de máquina em que uma coleção finita de propostas alternativas para a solução de um dado problema, denominadas componentes do ensemble, é empregada em conjunto na proposição de uma única solução para o problema (LIMA, 2004). Este paradigma mostrou que a habilidade de generalização do modelo pode ser melhorada por meio da combinação de classificadores.

Os experimentos com modelos combinados apresentaram melhores desempenho e generalização do que a utilização de um sistema decisório único. Esse fato tem motivado ainda mais o emprego dos ensembles em todas as linhas de aprendizado de máquina, sendo aplicados para classificação de padrões ou regressão.

A vantagem em utilizar esse paradigma é que a seleção do modelo apresenta melhores resultados do que os métodos tradicionais como, por exemplo, a validação cruzada (cross validation). Outro vantagem é a neutralização ou minimização da instabilidade inerente dos algoritmos de aprendizagem. Em contrapartida, modelos combinados são mais difíceis e custam mais para construir e analisar. Apesar do uso de ensembles normalmente apresentar bons resultados, não há garantia que isto ocorrerá sempre.

O sistema de classificação segue a abordagem de "dividir para conquistar", pois a fronteira de decisão, que separa os dados de diferentes classes, pode ser muito complexa e difícil de ser combinada. Portanto, a estratégia é dividir o conjunto de dados em porções menores e mais simples para o aprendizado de diferentes classificadores. Com isso, as fronteiras individuais de decisão dos sistemas de classificação tendem a ser mais simples e fáceis de serem apropriadamente combinadas pelo ensemble. Basicamente, o comitê aplica uma combinação nas fronteiras de decisão de cada classificador, a fim de encontrar uma fronteira que melhor represente os dados.

O sucesso de um comitê e a habilidade em corrigir erros dos seus componentes, depende da diversidade dos classificadores que o compõem, ou seja, cada classificador deve possuir erros distintos para instâncias de dados diferentes, para que assim o ensemble consiga obter uma melhor generalização. A ideia geral é construir muitos classificadores, treiná-los individualmente e combinar suas saídas de modo que o desempenho final seja melhor do que com um único classificador.

A construção do ensemble consiste na geração dos vários componentes (ou seja, os modelos independentes do comitê) e a combinação das saídas individuais dos componentes. Na maioria dos casos, para se ter um aumento no desempenho ou até mesmo a capacidade de mensurar tal desempenho, é necessário empregar a metodologia baseada em três passos: treinamento (geração dos componentes), seleção e combinação. No entanto, é necessário ter um conjunto de dados para geração dos componentes, outro para a seleção e combinação dos componentes e um conjunto para testar o desempenho do ensemble (LIMA, 2004).

As abordagens predominantes na geração de componentes são o bagging e boosting. Ambos tem o objetivo de tornar o mais diverso possível os subconjuntos de dados originados do conjunto de treinamento. O bagging, proposto por Breiman (1996), consiste no uso de diferentes subconjuntos (originados do conjunto de treinamento) de dados aleatoriamente criados com reposição (LIMA, 2004). Essa distinção aleatória entre os vários subconjuntos de treinamento, apesar de algumas amostras aparecerem mais de uma vez devido a reposição, confere diversidade aos modelos de classificação ou regressão que são obtidos a partir de cada um desses subconjuntos. Após o processo aleatória de diversidade dos subconjuntos de treinamento, cada modelo é treinado e suas saídas são combinadas por meio de voto majoritário com base em suas decisões.

No algoritmo boosting, proposto por Shapire (1990), a diversidade entre os conjuntos de treinamento não são gerados a partir de uma amostragem uniforme com reposição, como no caso do bagging. A probabilidade de escolha de uma amostra depende da contribuição desta para o erro de treinamento dos componentes já treinados, isto é, caso uma amostra não tenha sido corretamente classificada pelos componentes já gerados, a probabilidade de escolha desta aumenta em relação às demais amostras, quando do treinamento de novos componentes. Logo, esta amostra terá uma chance maior de ser escolhida para compor o conjunto de dados do próximo componente a ser gerado (LIMA, 2004). Portanto, esse processo de re-amostragem define os modelos de forma iterativa, sempre buscando formar o conjunto de treinamento mais informativo para cada classificador. Similarmente ao bagging, o boosting usa votação ou média para combinar as saídas de modelos individuais.

Além dos algoritmos para diversidade dos conjuntos de treinamento citadas anteriormente, tem-se também a versão mais genérica do algoritmo de boosting original, o algoritmo AdaBoost (ou Adaptive Boosting). Este, criado por Freund and Schapire em 1997, propõe que os conjuntos de treinamento são re-amostrados de forma adaptativa, de tal modo que amostras que mais contribuem para o erro de treinamento dos componentes já treinados possuem maiores probabilidades de comporem o conjunto de treinamento a ser empregado na síntese do próximo componente (LIMA, 2004). Logo, percebe-se que o AdaBoost é igual ao algoritmo boosting, porém diferem em como criar o conjunto treinamento mais informativo durante o processo iterativo. Além disso, o AdaBoost gera um conjunto de hipóteses e as combina por meio da votação ponderada.

A votação ponderada associa pesos as instâncias e o AdaBoost altera a distribuição da amostra modificando esses pesos. O valor dos pesos indicam quando determinada instância será selecionado ou não. O AdaBoost aumenta os pesos das instâncias erroneamente classificadas, para que possam ser selecionadas e reutilizadas no treinamento do próximo componente. Já as instâncias corretamente previstas possuem pesos menores (LIMA, 2004). Portanto, esse funcionamento do AdaBoost evidencia sua diferença em relação ao boosting, onde o treinamento ocorre a partir dos erros remanescentes e não da nova distribuição amostral.

4 EXPERIMENTOS

Antes de realizar os experimentos do conjunto de dados passou por uma abordagem de pré-processamento, a fim de adequá-los aos algoritmos e para eliminação de atributos desnecessários. Utilizando a técnica "variance threshold", onde os atributos que apresentarem variação diferente do limite setado (aproximadamente 0.8) serão removidos. A ideia é que se determinado elemento não varia muito dentro de si, geralmente tem pouco poder de previsão. Logo, o conjunto de dados de 12 atributos passou a ter exatamente 8 atributos referentes as temperaturas do pasto e da sombra, sendo eles: "tbs-pasto","ur-pasto", "tgn-pasto", "tpo-pasto", "tbs-sombra", "ur-sombra", "tgn-sombra", "tpo-sombra"

Além disso, a base também passou por um processo de normalização, devido as diferentes escalas e natureza dos atributos da base de dados. Os dados faltosos ao acaso (por motivos desconhecidos) foram repostos utilizando o valor da moda correspondente ao atributo faltoso. Optou-se por repor os dados ausentes para evitar a perda de informações significativas da base de dados e, possivelmente, afetar o processo de aprendizado dos algoritmos.

Na realização dos experimentos, aplicou-se a técnica da validação cruzada (ou "cross validation", em inglês) para validação dos modelos. Na abordagem básica, esta técnica serve para avaliar a capacidade de generalização de determinado modelo, a partir do conjunto de dados. Existem alguns métodos de validação cruzada, mas o utilizado para este artigo é o método k-fold, que consiste em dividir o conjunto total de dados em k subconjuntos disjuntos, onde k-1 das partições são utilizadas como treino e o restante é utilizada para validação do modelo.

Observando as tabelas com os resultados de cada método, percebe-se que há igualdade entre os valores de "recall"e "precision", isto se deve a taxa de falsos positivos e falsos negativos serem exatamente iguais. Uma possível explicação para esse fenômeno, como trata-se de um problema multiclasse, é o fato de que uma classificação incorreta de uma determinada classe "c1", onde o real valor era "c2", resulta em um falso positivo computado a c1 e um falso negativo computado a c2. Portanto, na soma de todos os falsos negativos e positivos será igual.

4.1 CARACTERÍSTICAS DOS DADOS

A base de dados selecionada para este artigo possui cerca de 287 instâncias de dados coletadas pelo estudante do Programa de Pós-graduação em Engenharia Agrícola Ambiental da UFRPE, Pedro Henrique Dias Batista, na Fazenda Roçadinho, localizada no município de Garanhuns, Pernambuco. Os dados usados para o treinamento dos classificadores são referentes a 24 horas de observação dos movimentos de uma única vaca, chamada Cassandra, e consistem em variáveis geográficas e meteorológicas. A figura 1 representa um exemplo de uma instância da base de dados.

tbs-pasto	ur-pasto	tgn-pasto	tpo-pasto	tbs-sombra	ur-sombra	tgn-sombra	tpo-sombra	classe
18.485	82.225	18.200	15.359	18.081	87.919	32.846	16.069	comendo
18.461	82.303	18.200	15.349	18.129	87.859	32.742	16.105	comendo

Figura 1: Instância aleatória da base de dados.

Na figura acima, observa-se o conjunto de variáveis meteorológicas consideradas para classificação. Estas variáveis, dependendo dos seus valores, remetem em que situação a vaca encontra-se. No caso das temperaturas no pasto estarem muito altas, as vacas irão procurar refúgio na sombra para controlar o equilíbrio térmico. Já a coluna "classes"apresenta valores inteiros positivos dentro de um intervalo de 0 a 4 que representam as seguintes classes nominais: andando(0), bebendo(1), comendo(2), ócio(3) e ruminando(4). Estas ações são justamente as características das saídas dos classificadores utilizados no desenvolvimento deste artigo.

4.2 MÉTRICAS ADOTADAS

A base de dados consistem 5 instâncias da classe 0, 4 instâncias da classe 1, 108 instâncias da classe 2, 103 instâncias da classe 3 e 68 instâncias da classe 4. Logo, que a base de dados está desbalanceada, pois observa-se que as classes bebendo e andando apresentam um número muito menor de ocorrências que as demais. Portanto, a base de dados encontra-se desbalanceada e por este motivo a métrica f-measure é aplicada para a seleção de modelos. A f-measure ou F1 score é uma medida de precisão de um teste e é definida como a média harmônica ponderada da precisão e da cobertura (ou recall) do teste. Essa métrica é representa pela equação a seguir.

$$f - measure = 2 \times \frac{precision \times recall}{precision + recall} \tag{54}$$

Na fórmula acima é possível observar duas novas métricas: precision (em português, precisão) e recall (em português, cobertura). A precisão indica o quão efetivamente correta estão as instâncias que foram classificadas corretamente pelo classificador. Já a cobertura é a frequência em que o classificador encontra os exemplos de uma classe, ou seja, determina o quão frequente se classifica uma instância de determinada classe, quando realmente ela é. Em outras palavras, o recall determina a capacidade do modelo em encontrar todos os casos relevantes dentro de um conjunto de dados. As equações são usadas para calcular a precisão e a cobertura, respectivamente.

$$Precisão = \frac{VerdadeirosPositivos(TP)}{VerdadeirosPositivos(TP) + FalsosPositivos(FP)} \tag{55}$$

$$Cobertura = \frac{VerdadeirosPositivos(TP)}{VerdadeirosPositivos(TP) + FalsosNegativos(NP)} \tag{56}$$

4.3 NAIVE BAYES LEARNING

O Naive Bayes é o único classificador deste artigo que não possui hiperparâmetros. Mesmo assim, durante o treino dividiu-se o conjunto de dados em duas partições disjuntas, assim como na técnica de validação cruzada, sendo elas: treino e teste, onde 20% do conjunto de dados original foram destinados ao conjunto de teste e 80% ao conjunto de treinamento. Nos resultados da tabela 1 é possível observar os valores de "recall", "precision"e "F-measure".

MÉTRICA	RESULTADO
Recall (cobertura)	**0,6897**
Precision (precisão)	**0,6897**
F-measure	**0,6897**

Tabela 1: Resultados das métrica do Naive Bayes.

4.4 K-NEAREST NEIGHBORS (kNN)

No treinamento do classificador kNN o hiperparâmetro k é um inteiro com distribuição de probabilidade uniforme em um intervalo [1,30]. Este hiperparâmetro é responsável por determinar o número de instâncias selecionadas dos vizinhos mais próximos, ou seja, a classificação de uma nova instância é dada pela proximidade dela com os k vizinhos mais próximos, onde será classificada baseada na classe majoritária de seus vizinhos. Segue abaixo os resultados da seleção de modelo do KNN.

MÉTRICA	RESULTADO
Recall (cobertura)	**0,6552**
Precision (precisão)	**0,6552**
F-measure	**0,6552**

Tabela 2: Resultados das métricas do KNN.

HIPERPARÂMETRO	VALOR
k	**3**

Tabela 3: Resultados dos hiperparâmetros do melhor modelo selecionado do kNN.

4.5 ÁRVORE DE DECISÃO (AD)

No caso da árvore de decisão, os hiperparâmetros considerados foram: o número mínimo de instâncias para divisão (representado na tabela 4 como "min_samples_split"), sendo um número inteiro com distribuição de probabilidade uniforme em [1, 50]; número mínimo de instâncias em uma folha (representado na tabela 4 como "min_samples_leaf"), sendo um número real com distribuição de probabilidade uniforme em [0, 1] e o número máximo de atributos considerados para divisão (representado na tabela 4 como "max_features"), sendo um número real com distribuição de probabilidade em um intervalo de [0, 1].

MÉTRICA	RESULTADO
Recall (cobertura)	**0,7414**
Precision (precisão)	**0,7414**
F-measure	**0,7414**

Tabela 4: Resultados das métricas da árvore de decisão.

HIPERPARÂMETRO	VALOR
Número mínimo de instâncias para divisão	**2**
Número mínimo de instâncias em folha	**0,1**
Número máximo de atributos considerados para divisão	**1**

Tabela 5: Resultados dos hiperparâmetros do melhor modelo selecionado da árvore de decisão.

4.6 REDES NEURAIS ARTIFICAIS (RNA)

No treinamento das redes neurais, os hiperparâmetros considerados foram os seguintes: número de nós escondidos, sendo representado por um inteiro com distribuição de probabilidade uniforme em [1, 100]; taxa de momento, com valor fixado em 0.8; número de épocas, com valor fixado em 500; algoritmo de treinamento, sendo representado pelo gradiente descendente; fração do conjunto de validação é fixo em 10%; função de ativação definida como tangente hiperbólica (tanh) e por fim a parada prematura é definida como verdadeira.

MÉTRICA	RESULTADO
Recall (cobertura)	**0,6897**
Precision (precisão)	**0,6897**
F-measure	**0,6897**

Tabela 6: Resultados das métricas da rede neural.

HIPERPARÂMETRO	VALOR
Número de nós escondidos	**68**
Taxa de momento	**0,8**
Número de épocas	**500**
Algoritmo de treinamento	**0,5**
Fração do conjunto de validação	**0,1**
Função de ativação	**tanh**
Parada prematura	**True**

Tabela 7: Resultados dos hiperparâmetros do melhor modelo selecionado da rede neural.

4.7 SUPPORT VECTOR MACHINE (SVM)

No caso do SVM, os hiperparâmetros considerados para a seleção de modelos foram: C, representado por valor exponencial com escala 10^5 e valor mínimo de 10^-7; kernel, sendo representado por uma distribuição categória em [linear, polinomial, rbf, sigmoide]; gamma (apenas

para kernel rbf e sigmoide), com distribuição uniforme em $[10^{-}7, 10]$ e grau (ou "degree", em inglês), representado por um número inteiro com distribuição uniforme em [2,4].

MÉTRICA	RESULTADO
Recall (cobertura)	**0,7069**
Precision (precisão)	**0,7069**
F-measure	**0,7069**

Tabela 8: Resultados das métricas do SVM.

HIPERPARÂMETRO	VALOR
kernel	**rbf**
degree	**2**
C	**3**

Tabela 9: Resultados dos hiperparâmetros do melhor modelo selecionado do SVM.

4.8 ENSEMBLE

Os resultados até aqui foram satisfatórios, porém utilizar o paradigma ensemble de aprendizado é uma das alternativas para melhorar o processo de classificação. Para isso, selecionou-se o algoritmo "Random Forest"(em português, "Floresta Aleatória- FA). Resumidamente, este algoritmo cria uma combinação (ensemble) de árvores de decisão. Segue abaixo os resultados obtidos nos experimentos a respeito do FA. Os hiperparâmetros considerados foram os mesmos da da árvore de decisão.

MÉTRICA	RESULTADO
Recall (cobertura)	**0,7759**
Precision (precisão)	**0,7759**
F-measure	**0,7759**

Tabela 10: Resultados das métricas do Random Forest.

HIPERPARÂMETRO	VALOR
Número máximo de atributos considerados para divisão	**0,5**
Número mínimo de instâncias em folha	**0,1**
Número máximo de atributos considerados para divisão	**2**

Tabela 11: Resultados dos hiperparâmetros do melhor modelo selecionado do Random Forest.

5 ANÁLISE DE RESULTADOS

Mediante os resultados expostos, pode-se inferir que o algoritmo Random Forest (RF) destacou-se com as melhores taxas de cobertura, precisão e f-measure em relação aos demais classificadores, sendo elas de aproximadamente 77,60%. Selecionou-se esse algoritmo devido

aos bons resultados apresentados pela árvore de decisão em relação ao SVM, RNA, KNN e Naive Bayes, sendo de aproximadamente 74,14%, enquanto que os demais,respectivamente, apresentaram: 70,69%, 68,97%, 65,52% e 68,97%. Indicando que o conjunto de treinamento mesmo em desequilíbrio e com múltiplas classes conseguiu obter resultados razoáveis no geral.

Além dos hiperparâmetros listados na tabela 11, há o hiperparâmetro chamado "n_estimator"(número de estimadores), que indica o número de árvores de decisão construídas pelo algoritmo antes de tomar uma votação ou fazer uma média de predições. Em geral, uma quantidade elevada de árvores aumenta a performance e torna as predições mais estáveis, mas também torna a computação mais lenta. Para este trabalho considerou-se um número total de 100 estimadores.

Outro hiperparâmetro importante é o número mínimo de instâncias em uma folha, ou seja, especifica o número mínimo de amostras necessárias para dividir um nó interno. Observando os resultados da tabela 11, pode-se inferir que determinado nó interno será divido apenas se tiver um número mínimo de amostras igual a 2. Porém, o valor em ponto flutuante do "Número mínimo de instâncias em folha"de 0,1 indica que o valor será uma fração do número total de amostras do conjunto de treino, logo será um número próximo de 28 (pois, o número de amostras é de 287). Mas, para concretizar a divisão é necessário olhar para o valor do número máximo de atributos, que por ser ponto flutuante (observado na tabela 11), será representado por um inteiro correspondente a metade dos atributos da base de dados.

Portanto, como o Random Forest divide o conjunto de dados em subconjuntos e utiliza-os para treinar individualmente os estimadores (as árvores de decisão), pode ocorrer das instâncias mais raras da base de dados ("bebendo"e "andando") não estarem presentes como nodo folha das árvores e, consequentemente, possuirá 0 instâncias classificadas como corretas. Com isso, é necessário analisar o número de instâncias que foram classificadas corretamente e incorretamente. Para isso, analisa-se a matriz de confusão descrita na tabela 12.

	andando	**bebendo**	**comendo**	**ócio**	**ruminando**
andando	**0**	**0**	**1**	**0**	**0**
bebendo	**0**	**0**	**1**	**0**	**0**
comendo	**0**	**0**	**34**	**4**	**2**
ócio	**0**	**0**	**5**	**21**	**9**
ruminando	**0**	**0**	**1**	**0**	**18**

Tabela 12: Matriz de consfusão gerada do treinamento do Random Forest.

Analisando a tabela 12, observa-se que as classes "andando"e "bebendo"possuem apenas duas instância selecionada, uma para cada, e classificadas incorretamente como instâncias da classe majoritária "comendo". Portanto, por estas serem raras e possuírem poucas instâncias, considera-se um erro de 100% para ambas. Logo, o comitê de classificadores Random Forest, mesmo apresentando os melhores resultados de f-measure não conseguiu se adequar a todas as classes. Porém, desconsiderando as classes raras (atípicas) o algoritmo mostrou bons resultados, classificando corretamente a maioria das ocorrências das classes: "comendo", "ócio"e

"ruminando". Não obstante, os demais classificadores também não conseguiram representar as classes atípicas.

Conclui-se que o Random Forest, apesar de conseguir um bom f-measure erro 100% para as classes "bebendo"e "andando". Mesmo aplicando o paradigma do ensemble e o seu fator de diversidade para formação dos subconjuntos de dados, extraiu-se cerca de 1 instância de 5 da classe "andando"e 1 instância de 4 da classe "bebendo". Essas aparições atípicas fizeram afetaram o desempenho do Random Forest, mas mesmo assim fez com que o f-measure permanecesse maior que os demais classificadores. Logo, isso justifica o percentual de f-measure do algoritmo.

Além do processo de combinação entre os modelos, o Random Forest gera para cada árvore interna um número aleatório de atributos, justamente por conta do paradigma de ensemble em tornar o mais diversificado possível os subconjuntos de dados. Essa estratégia conseguiu selecionar duas amostras atípicas da base de dados. Esse algoritmo consegue lidar bem com bases que apresentam muitos atributos e poucos exemplos, como é o caso deste artigo. Além disso, ele consegue tem um bom desempenho preditivo mesmo quando a maioria das variáveis preditivas são ruídos.

6 CONCLUSÃO

O processo de seleção de modelos e análise de resultados é fundamental para determinar a viabilidade de aplicação de um modelo de aprendizagem. Identificar o melhor algoritmo de aprendizado de máquina para determinado problema é uma tarefa árdua. O presente trabalho tem a finalidade de apresentar o modelo com a maior medida f-measure, ou seja, o de acurácia mais relevante.

Sendo assim, os resultados expostos da análise comparativa entre os classificadores apresentaram um nível satisfatório, diante de uma base de dados desbalanceada. Analisando os resultados, pode-se inferir que o algoritmo Random Forest obteve o melhor nível de f-measure que os demais modelos, sendo considerado o modelo o mais viável para classificar o comportamento das vacas de um rebanho com um f-measure de aproximadamente 77,60%. Detalhe, este artigo considera apenas o f-measure como método avaliativo para seleção do melhor classificador, desconsiderando um teste de hipotése robusto.

Como possíveis trabalhos futuros, pode-se sugerir a implementação de um sistema IoT para fazer a coleta dos dados das vacas de um determinado rebanho, e implantação do modelo Random Forest treinado neste artigo para atuar como classificador desses dados. Além disso, para futuras melhorias no processo de aprendizagem, sugere-se uma atualização no conjunto de dados (ou seja, um período maior de observações), com o intuito de reduzir o desequilíbrio das instâncias (em especial as que possuem os rótulos: "bebendo"e "andando") e aumentar a capacidade de generalização do modelo de aprendizagem.

REFERÊNCIAS

ERVILHA, Gabriel Teixeira. EFICIÊNCIA E DIFUSÃO DE TECNOLOGIA NA PRODUÇÃO DE LEITE EM MINAS GERAIS. 2014. 98 f. Dissertação (Mestrado) - Curso de Programa de Pós-graduação em Economia, Universidade Federal de Viçosa, Viçosa, 2014. Disponível em: <http://www.locus.ufv.br/bitstream/handle/123456789/3290/texto%20completo.pdf?sequence=1>. Acesso em: 08 abr. 2019.

GERON, Luiz Juliano Valério et al. Comportamento ingestivo de novilhas Nelore em pastejo recebendo suplemento a base de própolis ou monensina sódica. Semina: Ciências Agrárias, [s.l.], v. 35, n. 4, p.2047, 27 ago. 2014. Universidade Estadual de Londrina. 10.5433/1679-0359.2014v35n4p2047. Disponível em: <http://www.uel.br/revistas/uel/index.php/semagrarias/article/view/13880/14862>. Acesso em: 11 abr. 2019.

ROSA, Marcelo Simão da et al. Boas práticas de manejo:ordenha. Jaboticabal: Funep, 2009. 43 p. Disponível em: <http://www.agricultura.gov.br/assuntos/boas-praticas-e-bem-estar-animal/arquivos-publicacoes-bem-estar-animal/ordenha.pdf>. Acesso em: 11 abr. 2019.

NOBREGA, Luis et al. Animal monitoring based on IoT technologies. 2018 Iot Vertical And Topical Summit On Agriculture - Tuscany (iot Tuscany), [s.l.], p.1-5, maio 2018. IEEE. Disponível em: http://dx.doi.org/10.1109/iot-tuscany.2018.8373045. Acesso em: 21 maio 2019

WILLIAMS, M.l. et al. A novel behavioral model of the pasture-based dairy cow from GPS data using data mining and machine learning techniques. Journal Of Dairy Science, [s.l.], v. 99, n. 3, p.2063-2075, mar. 2016. American Dairy Science Association. 10.3168/jds.2015-10254. Disponível em: <https://www.journalofdairyscience.org/article/S0022-0302(16)00066-7/fulltext>. Acesso em: 21 maio 2019.

DUTTA, Ritaban et al. Dynamic cattle behavioural classification using supervised ensemble classifiers. Computers And Electronics In Agriculture, [s.l.], v. 111, p.18-28, fev. 2015. Elsevier BV. 10.1016/j.compag.2014.12.002. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168169914003123?via%3Dihub>. Acesso em: 22 maio 2019.

GONÇALVES, Thiago. Teorema de Bayes: o que é e qual sua aplicação?: Aprenda o que é o Teorema de Bayes, sua importância para a probabilidade condicional e suas aplicações. 2018. Leia mais em: https://www.voitto.com.br/blog/artigo/teorema-de-bayes. Disponível em: <https://www.voitto.com.br/blog/artigo/teorema-de-bayes>. Acesso em: 03 jun. 2019.

ZHANG, Harry. The Optimality of Naive Bayes. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference. University of New Brunswick, New Brunswick, 2004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.483.2183>. Acesso em: 04 jun. 2019.

OGURI, Pedro. Aprendizado de Máquina para o Problema de Sentiment Classification. 2006. 52 f. Tese (Doutorado) - Curso de Informática, Departamento de Informática do Centro Técnico Científico da Puc - Rio, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006. Disponível em: <https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=

resultado&nrSeq=9947@1>. Acesso em: 03 jun. 2019.

KOGA, Marcelo Li. Classificadores Bayesianos: Aplicados a análise sintática da língua portuguesa. In: São Paulo, 2011. p. 1 - 12. Disponível em: <http://stoa.usp.br/mlk/files/2741/17299/Classificadores_Bayesianos_relatorio.pdf>. Acesso em: 04 jun. 2019.

PARDO, Thiago Alexandre Salgueiro; NUNES, Maria das Graças Volpe. Aprendizado Bayesiano Aplicado ao Processamento de Línguas Naturais. Série de Relatórios Técnicos do Instituto de Ciências Matemáticas e de Computação - ICMC, Universidade de São Paulo, São Carlos: 2002. 26 p. Disponível em: <http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_180.pdf>. Acesso em: 04 jun. 2019.

HUANG, Olli. Applying Multinomial Naive Bayes to NLP Problems: A Practical Explanation. 2017. Disponível em: <https://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf>. Acesso em: 04 jun. 2019.

GUO, Gongde et al. KNN Model-Based Approach in Classification. On The Move To Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, [s.l.], v. 2888, p.986-996, 2003. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-540-39964-3_62. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.815&rep=rep1&type=pdf>. Acesso em: 05 jun. 2019.

DINIZ, Fabio Abrantes; SILVA, Thiago Reis da; ALENCAR, Francisco Eduardo Silva. Um estudo empírico de um sistema de reconhecimento facial utilizando o classificador KNN. Revista Brasileira de Computação Aplicada, [s.l.], v. 8, n. 1, p.50-63, 30 abr. 2016. UPF Editora. http://dx.doi.org/10.5335/rbca.2015.5227. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-540-39964-3_62>. Acesso em: 05 jun. 2019.

LIMA, Edirlei Soares de. INF 1771 – Inteligência Artificial: Rio de Janeiro: Edirlei Soares de Lima, 2011. 13 slides, color. Disponível em: <http://edirlei.3dgb.com.br/aulas/ia_2011_2/IA_Aula_14_KNN.pdf>. Acesso em: 05 jun. 2019.

CAMPOS, Raphael. Árvores de Decisão. 2017. Disponível em: <https://medium.com/machine-learning-beyond-deep-learning/%C3%A1rvores-de-decis%C3%A3o-3f52f6420b69>. Acesso em: 05 jun. 2019.

VON ZUBEN, Fernando J.; ATTUX, Romis R. F.. Árvores de Decisão. 2019. Disponível em: <http://www.dca.fee.unicamp.br/~vonzuben/courses/ia004.html>. Acesso em: 05 jun. 2019.

TATIBANA, Cassia Yuri; KAETSU, Deisi Yuki. Uma Introdução As Redes Neurais. 2004. Disponível em: <https://azdoc.tips/documents/uma-introduao-as-redes-neurais-tatibana-cassia-5c16ed4bb149b>. Acesso em: 06 jun. 2019.

TOGURO, Mary. O Modelo de McCulloch e Pitts. 2010. Disponível em: <https://th4w.wordpress.com/2010/09/12/o-modelo-de-mcculloch-e-pitts/>. Acesso em: 07 jun. 2019.

GANDHI, Rohith. Support Vector Machine - Introduction to Machine Learning Algorithms. 2018. Disponível em: <https://towardsdatascience.com/support-vector-machine-introduction-

to-machine-learning-algorithms-934a444fca47>. Acesso em: 13 jun. 2019.

KOWALCZYK, Alexandre. Support Vector Machines Succinctly. Morrisville: Syncfusion, 2017. Disponível em: <http://jermmy.xyz/images/2017-12-23/support_vector_machines_succinctly.pdf>. Acesso em: 13 jun. 2019.

LORENA, Ana Carolina; CARVALHO, André C. P. L. F. de. Introdução ás Máquinas de Vetores Suporte (Support Vector Machines). São Carlos: 2003. 66 f. Disponível em: <http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_192.pdf>. Acesso em: 13 jun. 2019.

IYODA, Eduardo Masato. INTELIGÊNCIA COMPUTACIONAL NO PROJETO AUTOMÁTICO DE REDES NEURAIS HÍBRIDAS E REDES NEUROFUZZY HETEROGÊNEAS. 2000. 166 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Universidade Estadual de Campinas, Campinas, 2000. Cap. 2. Disponível em: <http://www.dca.fee.unicamp.br/~vonzuben/research/emi_mest.htm>. Acesso em: 15 jun. 2019.

LIMA, Clodoaldo Aparecido de Moraes. Comitê de Máquinas: Uma Abordagem Unificada Empregando Máquinas de Vetores-Suporte. 2004. 378 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Universidade Estadual de Campinas, Campinas, 2004. Disponível em: <ftp://vm1-dca.fee.unicamp.br/pub/docs/vonzuben/theses/moraes_dout/tese_dout.pdf>. Acesso em: 16 jun. 2019.//

FACURE, Matheus. Funções de Ativação. 2017. Disponível em: <https://matheusfacure.github.io/2017/07/12/activ-func/#fun-act>. Acesso em: 15 jun. 2019.

ACADEMY, Data Science. Deep Learning Book, 2019. Disponível em: <http://deeplearningbook.com.br/funcao-de-ativacao/>. Acesso em: 15 jun. 2019.

ROQUE, Antonio. A Adaline. 2016. Disponível em: <https://docplayer.com.br/11667544-5945851-1-psicologia-conexionista-antonio-roque-aula-6-a-adaline.html>. Acesso em: 15 jun. 2019.