INSTITUTO FEDERAL DE SANTA CATARINA

MÁRIO ANDRÉ LEHMKUHL DE ABREU

Estudo de Técnicas de Aprendizado de Máquina em Reconhecimento de Voz para Detecção de Gênero

São José - SC

Agosto/2022

ESTUDO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA EM RECONHECIMENTO DE VOZ PARA DETECÇÃO DE GÊNERO

Monografia apresentada ao Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Prof. Elen Macedo Lobato, Dra.

Coorientador: Prof. Ramon Mayor Martins

São José - SC

Agosto/2022

MÁRIO ANDRÉ LEHMKUHL DE ABREU

ESTUDO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA EM RECONHECIMENTO DE VOZ PARA DETECÇÃO DE GÊNERO

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 29 de Agosto de 2022:

Prof. Elen Macedo Lobato, Dra.
Orientadora
Instituto Federal de Santa Catarina
Campus São José

Prof. Ramon Mayor Martins, Me.
Co-orientador
Instituto Federal de Santa Catarina
Campus São José

Prof. Mário de Noronha Neto, Dr.
Instituto Federal de Santa Catarina
Campus São José

Prof. Jorge Henrique Busatto Casagrande, Dr.
Instituto Federal de Santa Catarina
Campus São José

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus pela vida e saúde que possuo. Aos meus pais, irmão e irmã, e familiares que sempre me apoiaram e me incentivaram. A todos os professores, colegas e amigos que contribuíram durante todo o curso, e a minha orientadora e co-orientador, que me ajudaram e me incentivaram na elaboração desse trabalho.

O lucro dos nossos estudos
é tornarmo-nos melhores e mais sábios.
Michel de Montaigne

RESUMO

Com os avanços tecnológicos ocorridos nos últimos anos, o mundo vem passando por transformações, sendo notável cada vez mais a interação dos seres humanos com aplicações computacionais. Assistentes pessoais ativados por voz, carros autônomos, reconhecimento facial, marcação automática em fotos de redes sociais entre outras aplicações já são uma realidade no cotidiano da sociedade. Para implementar essas interações artificiais, é usada uma técnica que cresceu em popularidade na última década, chamada aprendizado de máquina (Machine Learning). Esta tecnologia é uma área da inteligência artificial (IA) que investiga como as máquinas podem aprender através da extração de característica a partir de um conjunto de dados. Através disso, várias aplicações podem ser desenvolvidas em vários cenários diferentes, como por exemplo, as citadas acima. Um que vem ganhando bastante presença no dia a dia é o reconhecimento de voz, que é integrado aos assistentes virtuais, e seu uso reflete em maior usabilidade ao realizar ações sem intervenção manual. Ele também oferece maior acessibilidade para aqueles com limitações motoras. No desenvolvimento desse recurso, uma vez que existem muitas variações da fala humana, seu reconhecimento pode ser feito entre vários aspectos, sendo um deles, o de identificação do gênero da voz, para uso em assistentes virtuais. Que ao reconhecerem o gênero do usuário, podem proporcionar mais interatividade, oferecendo determinados serviços para tornar a experiência do usuário mais envolvente. No entanto, para que a interatividade seja aceitável, a confiabilidade do reconhecimento deve estar o mais exata possível. Desse modo, esse trabalho teve como objetivo analisar e comparar o desempenho de algumas técnicas mais utilizadas no aprendizado de máquina para o reconhecimento do gênero da voz. Com o propósito de verificar qual técnica apresenta o melhor reconhecimento. Afim de servir como uma contribuição científica para informar qual técnica usar, a quem pretende implementar este tipo de reconhecimento em uma aplicação, poupando tempo no processo. Nesse desenvolvimento, toda a análise foi realizada em uma base de dados de vozes contendo os gêneros feminino e masculino disponível na web, e os resultados obtidos foram avaliados em termos de acurácia, sensibilidade, especificidade e eficiência.

Palavras-chave: Reconhecimento de voz, Reconhecimento de padrões, Inteligencia artificial, Aprendizado de máquina.

ABSTRACT

With the technological advances that have occurred in recent years, the world has been undergoing transformations, and the interaction of human beings with computer applications is increasingly remarkable. Voice-activated personal assistants, autonomous cars, facial recognition, automatic tagging in social network photos, among other applications, are already a reality in everyday society. To implement these artificial interactions, a technique that has grown in popularity in the last decade is used, called Machine Learning. This technology is an area of artificial intelligence (AI) that investigates how machines can learn through feature extraction from a dataset. Through this, several applications can be developed in various different scenarios, such as the ones mentioned above. One that is gaining a lot of presence in everyday life is voice recognition, which is integrated into virtual assistants, and its use reflects in greater usability by performing actions without manual intervention. It also offers greater accessibility for those with motor limitations. In developing this feature, since there are many variations of human speech, its recognition can be done among several aspects, one of them being the identification of the gender of the voice, for use in virtual assistants. By recognizing the user's gender, they can provide more interactivity, offering certain services to make the user experience more engaging. However, for the interactivity to be acceptable, the reliability of the recognition must be as accurate as possible. Thus, this work aimed to analyze and compare the performance of some of the most commonly used techniques in machine learning for voice gender recognition. The purpose was to verify which technique provides the best recognition. In order to serve as a scientific contribution to inform which technique to use, to those who intend to implement this type of recognition in an application, saving time in the process. In this development, all the analysis was performed on a database of voices containing the female and male genders available on the web, and the results obtained were evaluated in terms of accuracy, sensitivity, specificity and efficiency.

Keywords: Speech recognition, Pattern recognition, Artificial intelligence, Machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Com os avanços tecnológicos ocorridos nos últimos anos, o mundo vem passando por transformações, sendo notável cada vez mais a interação dos seres humanos com aplicações computacionais. Assistentes pessoais controladas por voz, carros autônomos, reconhecimento facial, marcação automática em fotos de redes sociais entre outras aplicações já são uma realidade. Desses exemplos, o que os tornam em comum é a tecnologia que os mesmos utilizam para realizar esses resultados, denominada Aprendizado de Máquina (*Machine Learning*) e que na última década vem ganhando bastante destaque.

O Aprendizado de Máquina representa o uso de algoritmos para extrair informações de dados brutos e representá-los por meio de algum tipo de modelo matemático. Através desse modelo, conclusões são feitas a partir de outros conjuntos de dados. (DATA SCIENCE ACADEMY, 2019a). Uma grande variedade de algoritmos diferentes são usados com a capacidade de classificar conjuntos de dados. O termo classificação é definido como o processo de atribuir a um determinado dado, o rótulo da classe que ele pertence. Assim, as técnicas de aprendizado de máquina são usadas na indução, através de um conjunto de dados de um classificador, tendo a capacidade de prever a classe de novos dados quaisquer a partir do treino realizado do conjunto de dados utilizado. (TAKAKURA et al., 2018).

Através do seu uso, várias aplicações já foram desenvolvidas em cenários diferentes e que já fazem parte do dia a dia da sociedade, como na recomendação de conteúdo, onde serviços de streaming como Netflix, Spotify e Amazon Prime Video usam a tecnologia para moldar o catálogo de filmes, séries, podcasts e músicas de acordo com as escolhas do usuário. Em apps de transporte e geolocalização como Uber e Cabify, que a partir dos dados das corridas, o sistema identifica padrões e se adapta conforme as mudanças de comportamento ocorrem, resultando na identificação da melhor rota. Na detecção de fraudes, onde as empresas de cartão de crédito a usam na prevenção de fraudes no processamento de pagamentos online, por meio da detecção de comportamento fora do padrão, o que possibilita bloquear o cartão no exato momento que é realizada uma compra. E no processamento de linguagem natural, com as interfaces ativadas por voz, como Alexa da Amazom, Google Assistente da Google, Siri da Apple e Cortana da Microsoft, que realizam ações definidas em suas configurações, ao falar com essas interfaces, como por exemplo, perguntar "Como está o tempo hoje?", se obtendo como resposta a previsão do tempo naquele momento.

Nesses diferentes cenários, um que vem sendo bastante utilizado é o reconhecimento de voz, através das assistentes virtuais. Desde que a Apple apresentou a Siri aos usuários do iPhone em 2011, a tecnologia de voz cresceu e se tornou um recurso básico não apenas

em smartphones, mas em uma variedade de dispositivos com acesso à Internet. Sendo o mais notável, o alto-falante inteligente que incorporado com uma assistente virtual, rapidamente é encontrado em mais de uma em cada dez casas conectadas com Wi-Fi, tendo-se Amazon, Google e Apple competindo nesse cenário (ENGLESON, 2017). Além desses, pode-se também encontrá-lo em sistemas embarcados automotivos, através do painel multimídia, nos serviços de telemarketing com os bots, entre outros. Como sua principal característica é a facilidade e usabilidade na realização de ações nas atividades diárias dos usuários, não sendo necessário o uso de meios físicos. Além disso, oferece maior acessibilidade para pessoas com deficiência física, permitindo que elas realizem atividades diárias sem precisar de auxílio.

Como a fala humana possui muitas variações, quando é utilizada como meio para realizar uma ação específica, em sua interpretação podem ser analisados diversos fatores, como identificação do gênero, emoções, a faixa etária do usuário, entre outros. Em termos de identificação do gênero, esse recurso incluído nos assistentes virtuais, pode proporcionar mais interação com o usuário, pois ao reconhecer o gênero, a aplicação pode fornecer determinados serviços, tornando a experiência de uso mais agradável. No entanto, para que a interação seja aceitável, a confiabilidade do reconhecimento deve ser a mais precisa possível.

Pelo fato do sinal de voz carregar muita informação, não é possível analisá-lo diretamente. Assim para o desenvolvimento da função de reconhecimento de voz é utilizado um sistema de reconhecimento automático de voz (RAV), cujo objetivo é transformar um sinal de voz em uma sequencia de dados na qual uma máquina tomará decisões. Para isso são executados três tarefas: A primeira é o pré-processamento, que inclui a conversão analógico/digital, filtragem e extração de parâmetros característicos do sinal de voz; A segunda é a identificação da informação contida no sinal de voz; A terceira e última é a comunicação, que representa o envio da informação reconhecida à aplicação que tomará decisões (CIPRIANO, 2002).

Na extração de parâmetros característicos são utilizados técnicas para obtenção dos coeficientes que representam o sinal de voz, resultando em um vetor de coeficientes. As técnicas mais conhecidas disponíveis para isso são *Mel Frequency Cepstral Coefficients* (MFCC), *Linear Predictive Coding* (LPC) (Shrawankar; Thakare, 2010), e *Perceptual Linear Prediction Coefficients* (PLP) (Fan et al., 2012). A partir dos coeficientes extraído, para a tarefa de identificação, onde ocorre o reconhecimento de padrões são utilizadas técnicas como *Hidden Markov Model* (HMM)(YOUNG, 2008), *Dynamic Time Warping* (DTW) (MUDA; BEGAM; ELAMVAZUTHI, 2010), *Artificial Neural Networks* (ANN), *Support Vector Machines* (SVM), *Decision tree* (DT), *Random forests* (RF), *Naive Bayes* (NB), *K-Nearest Neighbors* (KNN), (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007) e mais recentemente *Deep Learning* com as técnicas *Recurrent Neural Network* (RNN) e

Convolutional Neural Network (CNN)(GOODFELLOW et al., 2016).

Como a fala humana é considerada de elevada complexidade, o que determina um resultado viável no uso do RAV são alguns fatores como: desempenho do sistema de captação do sinal de fala; capacidade do sistema de separar o sinal de fala de ruídos externos; capacidade do sistema de separar duas ou mais vozes misturadas em um único sinal; desempenho do sistema digitalizador; desempenho das técnicas de extração das propriedades acústicas dos sinais de fala; desempenho do modelo classificador e de reconhecimento de padrões; distâncias relevantes entre o sistema de captação do sinal de voz e do locutor e efetividade de técnicas dinâmicas, de normalização e detecção de voz ativa (MACHADO, 2016). Deste modo, escolher as melhores configurações em cada fator, é o que vai estabelecer os resultados mais satisfatórios.

Assim, em relação ao desempenho do modelo classificador, que envolvem o uso de técnicas de aprendizado de máquina, nesse trabalho foi feito um estudo com seis técnicas mais usadas desse contexto, sendo elas *Artificial Neural Networks (ANN)*, *Support Vector Machines (SVM)*, *Decision Trees (DT)*, *Random Forest (RF)*, *Naive Bayes (NB)* e *K-Nearest Neighbors (K-NN)*. Para verificar quais delas apresentam o melhor desempenho de reconhecimento, na identificação do gênero da voz. Servindo como um estudo, na escolha de qual técnica usar na etapa de identificação de um sistema Reconhecimento Automático de Voz (RAV).

1.1 Motivação

Aplicações com assistentes virtuais acionadas por voz, já são uma realidade cada vez mais notável no momento atual da sociedade, no qual, possuem como objetivo auxiliar os usuários na realização de tarefas usando como meio a voz. Utilizando a combinação de técnicas baseadas em Inteligência Artificial e Aprendizado de Máquina, uma aplicação com o reconhecimento de voz, recebe estímulos por voz, interpreta a fala e executa os comandos recebidos. Esses assistentes virtuais podem ser encontrados em smartphones e computadores, em sistemas embarcados automotivos, através do painel multimídia, nos serviços de telemarketing com os bots, entre outros. Como a fala humana possui inúmeras variações, na sua interpretação vários fatores podem ser analisados, como identificação de gênero, a faixa etária, sentimentos, entre outros. No que diz respeito, a identificação de gênero, esse recurso incorporado em determinados assistentes virtuais, podem proporcionar mais interatividade com o usuário, pois reconhecendo o gênero do usuário, a aplicação pode oferecer determinados serviços, tornando a experiencia de uso muito mais interessante. No entanto, para que a interatividade seja aceitável, a confiabilidade do reconhecimento deve ser a mais precisa possível. Assim, este trabalho teve como objetivo, apresentar um estudo sobre técnicas de aprendizado de máquina para o reconhecimento de gênero da voz, de modo a analisar qual técnica apresenta o melhor desempenho, servindo de referência na escolha de qual técnica usar ao implementar esse recurso em aplicações que precisem dessa característica.

1.2 Objetivo geral

Estudar técnicas de aprendizado de máquina para reconhecimento e classificação de voz por gênero, e verificar qual apresenta o melhor reconhecimento, com o propósito de uso em aplicações que precisem dessa característica.

1.3 Objetivos específicos

- Estudar e aplicar técnicas e algoritmos de aprendizado de máquina.

- Analisar o desempenho das técnicas a partir de um conjunto de dados pré-definido.

- Determinar, através dos resultados de testes, as técnicas que apresentem melhor desempenho de acordo com o objetivo geral do trabalho.

1.4 Organização do texto

A divisão deste documento foi realizada da seguinte forma. O Capítulo 1 apresenta o contexto da proposta, a motivação, objetivos gerais e específicos.

No Capítulo 2 é realizada uma revisão sobre a característica da fala, fundamentos sobre o processamento de voz, sistemas RAV, especificado as técnicas de aprendizado de máquinas utilizadas e o processo de validação usada pelas mesmas.

No Capítulo 3 é apresentado a base de dados escolhida e como foi configurado o algoritmo de treinamento das técnicas escolhidas.

No Capítulo 4 é exibido os resultados obtidos. Isto é, é mostrado a aplicação dos dados de treino sobre os modelos treinados, mostrando quais técnicas tiveram os melhores desempenhos. O resultado da aplicação dos dados de teste sobre os modelos de treino criados, apresentando a comparação dos resultados obtidos por meio de métricas de acurácia, sensibilidade, especificidade e eficiência. E por último, uma comparação dos resultados em relação ao número de amostras usadas no treino dos modelos.

Por último, no Capítulo 5 é apresentado a conclusão dos resultados obtidos e sugerido trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

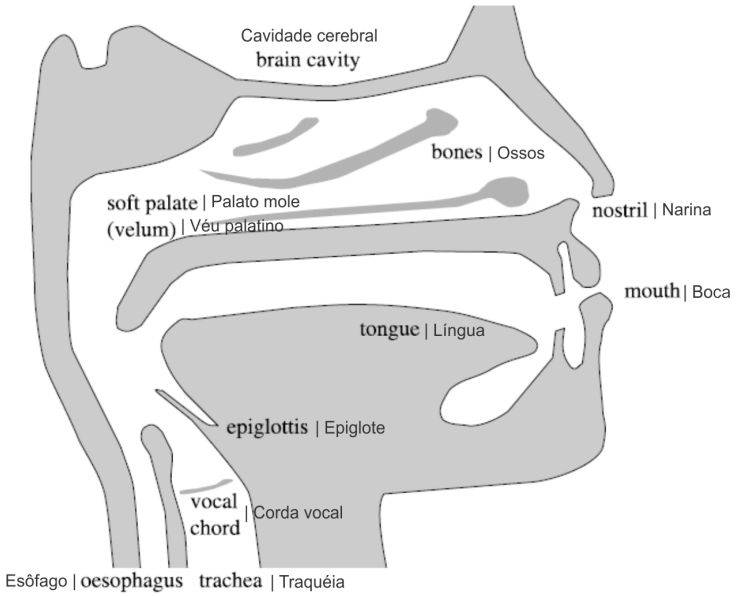
2.1 Característica da fala

Um fenômeno que existe desde o nascimento do ser humano é a sua voz, apresentado-se de várias maneiras, como risos, choro, grito e sons de fala. Ela é um dos meios de comunicação que um indivíduo utiliza para se expressar com o exterior e particularmente com seus semelhantes (GABANINI, 2003). A fala pelo ponto de vista de áudio não apresenta nada de especial, é simplesmente como qualquer outro som. No entanto, a partir do momento que o cérebro humano passa a ouvi-la, ele o interpreta de maneira diferenciada considerando-a como uma fala.

O som conhecido como fala tem seu inicio nos pulmões, onde os mesmos se contraem para expelir o ar que o transporta de uma distribuição de frequência aproximadamente gaussiana. O ar é pressionado pelo trato brônquico, passando por um conjunto de dobras musculares no topo da traqueia denominada cordas vocais, fazendo-as vibrar. O ar então chega na parte posterior da cavidade bucal seguindo um de dois caminhos possíveis para o exterior. O primeiro caminho é sobre e em torno da língua, passando pelos dentes e saindo pela boca. O segundo caminho é pela cavidade nasal, que quando o véu palatino esta fechado é o único caminho a se seguir. Na produção final do som a combinação de muitos fatores são necessárias como a potência pulmonar e modulação da pressão, redução na glote, tensão na cordas vocais, formato da boca e a posição da língua e dos dentes. A realização sincronizada das articulações responsáveis pela modulação da voz, desempenham a locução de fonemas, que são divididos em vogais e consoantes, gerando um conjunto de palavras que formam a mensagem a ser expressa pela fala (MCLOUGHLIN, 2009). Na figura 1 é mostrado um diagrama do aparelho de produção da fala, também chamado de cabeça humana .

O trato vocal que começa na glote e termina nos lábios é um tubo acústico variável e não uniforme, que tem a função de agir como um tubo ressonante com a finalidade de filtrar o conjunto de pulsos produzidos. No trato vocal um dos componente principais que o compõem são os articuladores, que representam as partes moveis como maxilar, véu palatino, língua e lábios. Sendo o véu palatino uma peça fundamental, atuando na abertura e fechamento da cavidade nasal, controlando o som produzido pelas narinas. Nas cordas vocais o movimento oscilatório acontece eu uma frequência fundamental, denomina *pitch*, que varia durante a fala e esta relacionada com a magnitude das variações de pressão sobre a glote, da tensão nas cordas vocais e da massa das bordas vibrantes, resultando na entonação da pronuncia (VALIATI, 2000).

Figura 1 – Diagrama de uma porção do trato vocal.



FONTE: (MCLOUGHLIN, 2009).

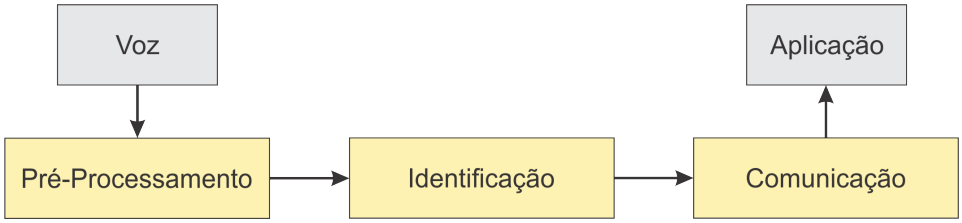
Os sinais de fala possuem peculiaridades que as diferenciam dos demais sinais de outros meios, mesmo as características espectrais e temporais sendo semelhantes. Isso ocorre através de um estruturado conjunto de sons contínuos que define conjuntos espectrais variantes de 20 Hz a 20 khz. Sendo sons nasais ou vozeados(vogais e as consoantes j, l m) que possuem um espectro discreto com uma frequência fundamental variantes de 100 Hz a 200Hz para homens e de 200 Hz a 400 Hz para mulheres. E sons não vozeados (consoantes f, s, p e dígrafo ch) que são formados pelo fluxo de ar na boca, modulados pelos maxilares, língua e lábios, e definem espectros contínuos (MACHADO, 2016).

Dessa forma é importante conhecer quais distorções prejudicam a qualidade e inteligibilidade do sinal de voz utilizado, com o objetivo de executar técnicas que consigam interpretar que o mesmo é um sinal de voz. Além de estimar em algumas situações o que foi enunciado, e também separar a voz de ruídos presentes durante a captação do sinal. Nesse sentido, a utilização de um sistema RAV é um recurso disponível, onde são usadas técnicas que tentam imitar o comportamento do reconhecimento da fala humana.

2.2 Sistemas de reconhecimento de voz

Um sistema de reconhecimento automático de voz (RAV) é um sistema que tem por objetivo transformar um sinal de voz em uma sequencia de dados no qual uma aplicação tomará decisões. No seu processo de execução são realizadas três etapas, sendo elas pré processamento, identificação e comunicação, como é mostrado na figura 2.

Figura 2 – Etapas de um Sistema de reconhecimento automático de voz (RAV).



FONTE: PRÓPRIO AUTOR.

Em cada uma dessas etapas, tarefas são realizadas, como descritas abaixo:

1. **Pré-processamento:** Ocorre a conversão do sinal de voz de analógico para digital (A/D), filtragem e extração da informação presente no sinal de voz. Na etapa de extração de informações do sinal, os valores resultantes, irão depender da técnica utilizada.

2. **Identificação:** Ocorre o reconhecimento do sinal através dos parâmetros extraídos. Esse reconhecimento é realizado por técnicas específicas, como por exemplo, as de aprendizado de máquina, HMM e DTW.

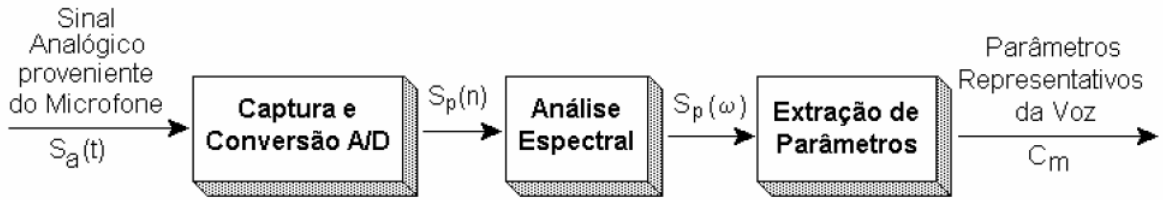
3. **Comunicação:** Ocorre o encaminhamento da informação reconhecida a aplicação que decidirá qual ação realizar (CIPRIANO, 2002).

Definindo-se as características de como o sinal de voz vai ser usado, o processo de preparação em cada tarefa do sistema RAV pode ser executada.

2.3 Etapas de pré processamento e extração de características

A primeira etapa de um sistema de reconhecimento automático de voz é o pré-processamento. Nela é feita a conversão do sinal de analógico para digital, a análise espectral e a extração das características acústicas, como mostrado na figura 3.

Figura 3 – Etapas do Pré-Processamento.



FONTE: (CIPRIANO, 2001).

O objetivo da etapa de pré-processamento do sinal é estabelecer um conjunto de parâmetros que contenha informações úteis, para serem usadas na etapa de identificação.

2.3.1 Conversão A/D do sinal de voz

Os sinais analógicos são aqueles em que sua amplitude varia continuamente com o tempo, isto é, ela pode assumir qualquer valor pertencente a um intervalo contínuo de valores. Já um sinal digital apresenta amplitudes dentro de um conjunto de valores finito que varia de forma discreta com o tempo. Trabalhar com dados digitais é mais indicado pois o seu processamento é mais eficiente e confiável que os dados analógicos (FERNANDES; PANAZIO, 2009).

A conversão analógico para digital ocorre da amostragem do sinal analógico $S_a(t)$, no período T segundos e da quantização das amostras para resultar no sinal digital $S(n) = S_a(n.T)$, onde $n = 0, 1, 2, ...$. Para que a informação do sinal se mantenha, o critério de *Nyquist* deve ser seguido, que diz que que a frequência de amostragem do sinal dever ser maior ou igual a duas vezes a sua frequência máxima ($f_s >= 2.f_{max}$). Para não ocorrer a sobreposição do espectro de $S_a(t)$, impossibilitando a recuperação fiel do sinal original (efeito aliassing), o sinal de voz dever ser primeiramente passado em um filtro analógico passa-baixas com frequência de corte menor ou igual a metade da frequência de amostragem ($f_c <= f_s/2$), eliminando frequência com valores maiores que a metade da frequência de amostragem. Concluído esse processo o sinal é passado em um conversor analógico para digital.

Com o sinal convertido para digital, ele é inserido em um filtro de pré-enfase, com função de transferência, mostrada na equação 2.1, com o objetivo de equalizar o espectro de voz e aprimorar o desempenho da análise espectral, que acontece na próxima etapa (CIPRIANO, 2001).

$$H_{pre}(z) = 1 - a_{pre}.z^1 \tag{2.1}$$

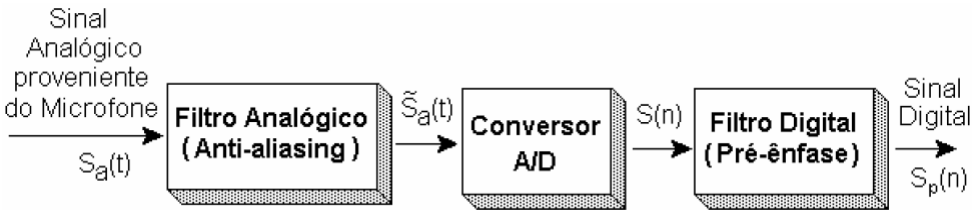
onde $0,9 \leq a_{pre} \leq 1$ é o coeficiente de pré-enfase.

A saída $S_p(n)$ do filtro de pré-enfase, tem relação com a entrada $S(n)$ por meio da equação 2.2 .

$$S_p(n) = S(n) - a_{pre}.S(n-1) \tag{2.2}$$

Na figura 4 é mostrado as etapas da conversão do sinal de voz analógico para digital.

Figura 4 – Etapas da conversão do sinal de voz analógico para digital.



FONTE: (CIPRIANO, 2001).

2.3.2 Análise espectral

A análise espectral é um processo que tem como objetivo converter o sinal de voz, que está no domínio do tempo, para o domínio da frequência. Esta conversão acontece pelo fato da audição e percepção humana estarem bem mais correlacionadas com a representação espectral do que na representação temporal do sinal de voz (NUNES et al., 1996).

Nas aplicações práticas de processamento de sinais em geral é necessário utilizar pequenas porções ou frames (quadros) do sinal. Em sinais em que a duração é muito curta isso não é necessário. Isso acontece pelo uso de técnicas de análise convencionais de sistemas lineares invariantes no tempo (estacionários). Desse modo é preciso selecionar um pedaço de sinal que possa ser minimamente assumido como estacionário. Assim o sinal é dividido em frames (quadros) de N amostras, sendo os frames vizinhos separados por M amostras. A divisão varia de 10 a 25 ms, pois o sinal é quase estacionário nessa faixa de valores.

Dessa forma, define-se um frame de voz como o produto de uma janela discreta $w(n)$ de tamanho L e terminando no tempo "l", com relação à sequencia de voz discreta (pré-enfatizada) $y(n)$, resultando na seleção de um pedaço do sinal pré-enfatizado, como mostrado na equação 2.3:

$$f(n) = y(n).w(l - n) \tag{2.3}$$

onde $f(n)$ é um frame do sinal pré-enfatizado $y(n)$, e $w(n)$ é a aplicação da janela (PETRY; ZANUZ; BARONE, 2000).

O sinal é dividido em frames através do janelamento. Executar o janelamento no domínio do tempo em resumo significa multiplicar o sinal pela função da janela utilizada. Como a multiplicação do domínio do tempo corresponde a convolução no domino da frequência, a aplicação da janela no domínio do tempo altera a forma do sinal no domínio da frequência.

Para dividir em frames são usadas janelas de Hamming. As janelas possuem períodos

um pouco maiores que os frames, formando uma região de sobreposição, com o objetivo de garantir que a variação dos parâmetros entre janelas vizinhas ocorra de forma gradual e que a análise das informações situadas nos extremos das janelas não seja prejudicada (SILVA, 2009). A função que representa a janela de Hamming é mostrada na equação 2.4.

$$\begin{aligned} w(n) &= 0,54 - 0,46\left(\frac{2\pi n}{N-1}\right), \quad n = 0, 1, ..., N-1 \\ w(n) &= 0, \quad \text{caso contrário} \end{aligned} \tag{2.4}$$

Em sistemas RAV dois métodos de análise espectral são os mais utilizados, sendo eles o método de análise espectral por banco de filtros através da transformada rápida de Fourier(FFT) e o método de análise espectral LPC(Linear Predictive Coding)(NUNES et al., 1996). Realizando-se a conversão do sinal para o domínio da frequência, o processo de extração dos parâmetros acústicos pode ser iniciado.

2.3.3 Extração das características

A extração de características do sinal de voz é definida como um processo de criação de vetores que carregam informações acústicas úteis que descrevem o conteúdo do sinal, denominado vetores acústicos. No processo de extração das propriedades acústicas do sinal são eliminadas várias fontes de informação, como segmentos que possuam somente ruídos ou ausência de som audível (normalmente regiões iniciais e finais de uma gravação), efeitos de periodicidade, amplitude do sinal de excitação, reverberação, componentes de ruído e frequências fundamentais.

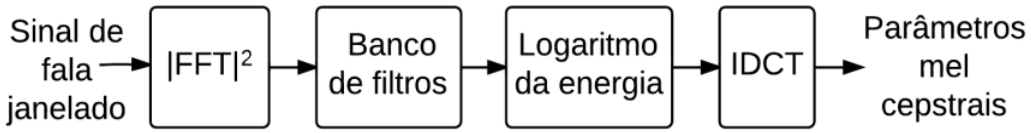
As técnicas mais usadas para a extração são *Mel Frequency Cepstral Coeffcients* (MFCC),*Linear Predictive Conding* (LPC) e *Perceptual linear Predictive Coefficients* (PLP)(MACHADO, 2016). Dessas três técnicas, o MFCC é frequentemente o mais utilizado, pois no processamento da fala oferece uma metodologia para a separação do sinal de excitação da resposta impulsiva do trato vocal (PETRY; ZANUZ; BARONE, 2000).

O MFCC é uma técnica de extração de características acústicas de sinais de voz, que trabalha no domínio da frequência, e utiliza uma escala chamada Mel, que corresponde a escala do ouvido humano. Os coeficientes mel-cepstrais são representações da parte real do cepstro de um janelamento em um período curto de tempo de sinais acústicos, obtidos de uma transformada rápida de Fourier (FFT) de um sinal. A distinção desses coeficientes com os coeficientes cepstrais reais, está no uso de uma escala logarítmica, que aproxima os coeficientes ao comportamento do aparelho auditivo humano (MACHADO, 2016).

No processo de análise mel-cepstral, deve-se seguir algumas sequências de passos, sendo a primeira a análise cepstral em cada janela, que corresponde inicialmente em calcular o quadrado do módulo da FFT das amostras pertencentes a janela de análise. No

próximo passo, filtrar o sinal resultante por um banco de filtros triangulares na escala Mel. No passo seguinte calcular o logaritmo da energia na saída dos filtros e por último calcular a IDCT (Inverse Discrete Cosine Transform). Na figura 5 é mostrado o diagrama de bloco que descreve esse processo.

Figura 5 – Diagrama de blocos usando banco de filtros para calcular os parâmetros mel-cepstrais .



FONTE: (MARTINS; YNOGUTI, 2014b).

A forma do banco de filtros está em discretizar o espectro de frequências de modo correspondente a como é processado os sinais no aparelho auditivo humano. Nos filtros as frequências centrais se alteram linearmente até a frequência de $1khz$, depois desse valor começam a crescer exponencialmente com um fator de 21/5 (MARTINS; YNOGUTI, 2014b). Na tabela 1 é mostrado essas frequências centrais dos filtros na escala Mel.

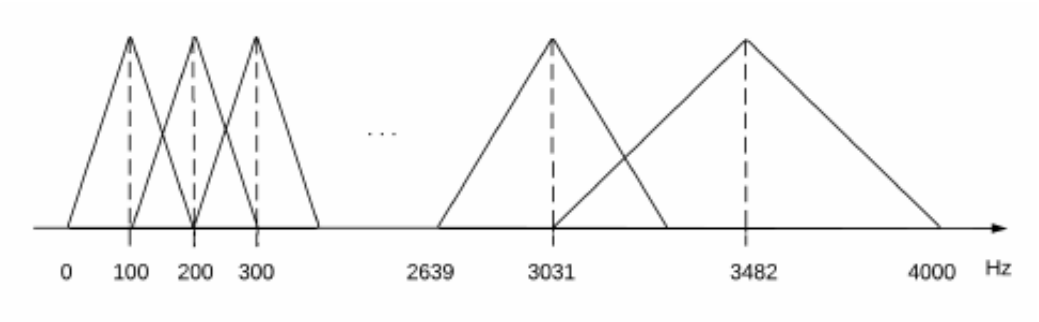
Tabela 1 – Frequências centrais dos filtros na escala Mel

Índice dos Filtros	**Frequência central (Hz)**	**Índice dos Filtros**	**Frequência central (Hz)**
1	100	13	1516
2	200	14	1741
3	300	15	2000
4	400	16	2297
5	500	17	2639
6	600	18	3031
7	700	19	3482
8	800	20	4000
9	900	21	4595
10	1000	22	5278
11	1149	23	6063
12	1320	24	6964

FONTE: (MARTINS; YNOGUTI, 2014a).

Na construção do banco de filtros na escala Mel, pode-se implementá-los através de filtros passa-banda triangulares, no qual o $k-ésimo$ filtro é definido na frequência central $f(k)$, onde sua banda se estende da frequência $f(k-1)$ á frequência $f(k+1)$, de modo que $f(k+1)$ não passe do valor de $fs/2$. Para um sinal amostrado a $8khz$ a implementações dos filtros ficaram como mostrado na figura 6. Neste exemplo percebe-se que seria utilizado 19 filtros, como pode ser visto olhando a tabela 1.

Figura 6 – Banco de filtros na escala mel para uma frequência de amostragem de 8 kHz.



FONTE: (MARTINS; YNOGUTI, 2014a).

O modelo matemático que descreve como obter o MFCC é mostrado na equação 2.5

$$c(n) = \sum_{k=1}^{K} log|(S_k)| \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right] \tag{2.5}$$

para $0 <= n < P$, onde $c(n)$ é o n-ésimo coeficiente mel-cepstral, P é o número de coeficientes mel-cepstrais extraídos, K é o número de filtros digitais utilizados e $S(k)$ é o sinal de saída do banco de filtros digitais (PETRY; ZANUZ; BARONE, 2000).

Estando concluído a geração do vetor de coeficientes, que representa as características acústicas do sinal de voz, o processo de reconhecimento pode ser iniciado na tarefa de identificação do sistema RAV utilizando técnicas de aprendizagem de máquina.

2.4 Aprendizagem de máquina (Machine Learning)

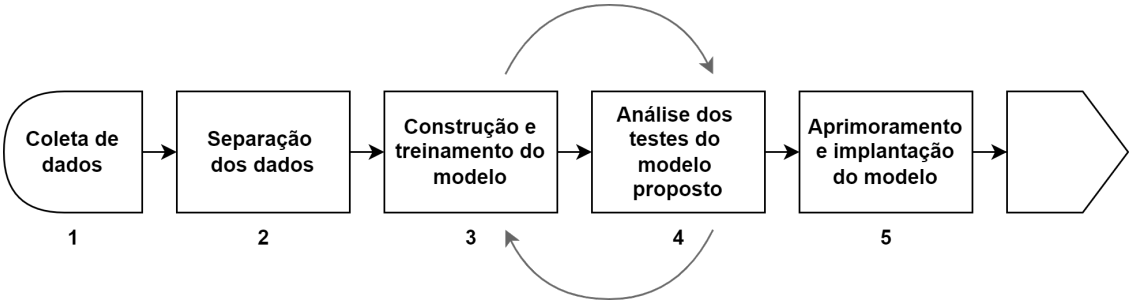
O aprendizado de máquina do inglês Machine Learning (ML) é uma área de pesquisa da inteligência artificial que estuda o desenvolvimento de métodos capazes de extrair conhecimento a partir de amostras de dados. Vários algoritmos são usados para criar classificadores para um conjunto de exemplos, sendo o termo *classificação* definido como o processo de atribuir a uma dada informação, o rótulo da classe a qual ela pertence (LORENA; CARVALHO, 2003). Desse modo, por meio de um conjunto de treinamento as técnicas de AM são usadas na indução de um classificador que consiga prever a classe de instâncias quaisquer no contexto em que ele foi treinado.

Nos últimos anos o avanço dessa área em conjuntos com outras áreas tecnológicas, está proporcionando que computadores tomem decisões sem estarem explicitamente programados, permitindo dessa forma que tarefas que por um bom tempo foram consideradas impossíveis de serem executadas por máquinas começassem a ser automatizadas. Alguns

exemplos disso são o reconhecimento de voz, a automação na área da robótica, carros autônomos, atendentes artificiais (bots), entre outros.

Na maioria dos projetos que usam o aprendizado de máquina, o fluxo de execução realizado no processo de aprendizado é mostrado na figura 7. Nela pode-se observar que ao realizar todas as etapas, se espera como resultado final um modelo satisfatório e eficaz. Na execução desse fluxo algumas etapas são realizadas como descrito abaixo:

Figura 7 – Fluxo de execução do processo de aprendizado de máquina.



FONTE: PRÓPRIO AUTOR.

- **Etapa 1 - Coleta dos dados:** Os dados que serão usados para o desenvolvimento do projeto são identificados e coletados.

- **Etapa 2 - Separação dos dados:** Os dados coletados passam por um processo de limpeza, isto é, alguns dados podem ser alterados para outros valores ou podem ser retirados caso necessário, para se adequarem na análise e serem utilizados. Além disso, efetua-se a divisão dos dados em duas partes, onde uma parte será usada para treinamento do modelo denominada conjunto de teste e a outra para os testes do modelo denominada conjunto de teste.

- **Etapa 3 - Construção e treinamento do modelo:** O modelo inicial é construído e um algoritmo ou método é selecionado. Esse é um modelo que parte de um conhecimento prévio dos dados, implicando na identificação da capacidade de assertividade do algoritmo escolhido. Dessa forma, faz-se o treinamento desse modelo com os dados já separados da etapa anterior.

- **Etapa 4 - Análise dos testes do modelo proposto:** Após o treinamento do modelo, e os dados separados para testes terem sido aplicados ao mesmo, uma análise dos resultados será feita. Em casos em que os resultados não são satisfatórios, é necessário voltar para a etapa 3 do fluxo do processo de aprendizado mostrado na figura 7.

- **Etapa 5 - Aprimoramento e implantação do modelo:** Representa que o modelo foi validado e aplicado, obtendo resultados adequados. Desse modo, o modelo já pode ser implantado e até aprimorado (ARRIGONI, 2018).

2.5 Técnicas de aprendizagem de máquina

Ao se trabalhar com o aprendizado de máquina, três paradigmas podem ser escolhidos para fazer a classificação dos dados, sendo eles, aprendizado supervisionado, não-supervisionado e por reforço.

No aprendizado supervisionado um conjunto de exemplos de treinamento é fornecido nos quais o rótulo (label) das classes designadas são conhecidos, representando que as entradas e as saídas dos dados são conhecidas. Desse modo, cada amostra no conjunto de dados é composto por um vetor de valores de características ou atributos e o rótulo da classe designada. O algoritmo então tem por objetivo criar um classificador que determine corretamente a classe de novos exemplos que ainda não possuam o rótulo da classe. Quando os rótulos das classes são discretos, ou seja, apresentam estados diferentes como por exemplo, saber se a voz é feminina ou masculina, se uma fruta é banana, maçã ou laranja e se pessoas estão doentes ou saudáveis pelos seus diagnósticos, esse tipo de análise é definido como classificação. Já se forem contínuos, como por exemplo, as medidas de peso e altura, esse tipo de cenário é uma regressão.

No aprendizado não-supervisionado não se conhece a saída, dessa modo os exemplos fornecidos são analisados e tenta-se observar se alguns deles podem ser agrupados de alguma forma. Ocorrendo os agrupamentos, é realizado uma análise para determinar o que cada agrupamento representa no contexto do problema que se está sendo analisado (MONARD; BARANAUSKAS, 2003).

No aprendizado por reforço, o processo ocorre através de recompensas ou não ao classificador, dependendo do desempenho obtido na aproximação da função desejada (LORENA; CARVALHO, 2003).

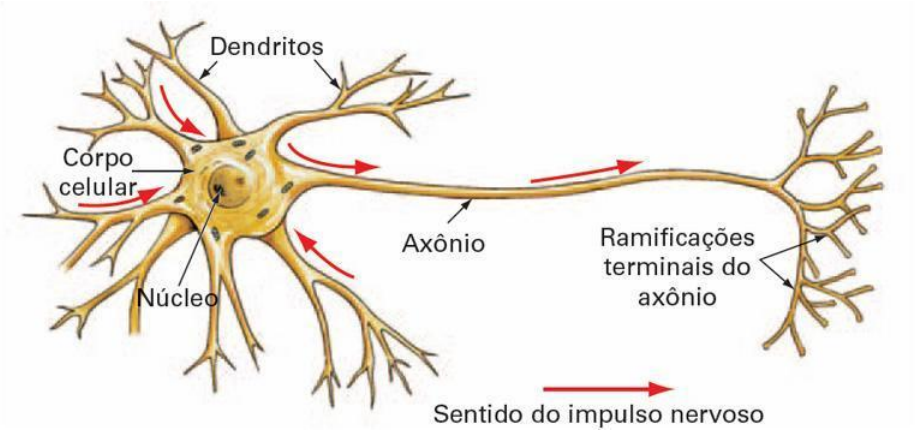
Para este trabalho, como os dados de entrada e saída são conhecidos, o aprendizado utilizado escolhido é o supervisionado. Além disso, como os rótulos das classes são discretos, isto é, as classes são as vozes femininas e masculinas, a análise feita é do tipo classificação. Assim para o processo de treinamento se escolheu técnicas de aprendizado supervisionado, que são apresentadas nas subseções a seguir.

2.5.1 Artificial Neural Networks (ANN)

A técnica *ANN* do português *Redes Neurais Artificiais* foi desenvolvida a partir do neurônio biológico humano. O neurônio é uma célula do cérebro humano, especializado

na transmissão de informações, pois possui propriedades de excitabilidade e condução de mensagens nervosas. Sua estrutura é constituída por 3 partes principais: o corpo celular ou a soma, do qual saem algumas ramificações chamadas de dendritos, e por uma outra ramificação, entretanto mais extensa, chamada de axônio. Nas extremidades dos axônios estão os nervos terminais, onde ocorre a transmissão das informações para outros neurônios, conhecida como sinapse. Na figura 8 é mostrado a estrutura do neurônio humano.

Figura 8 – Visão Simplificada do neurônio biológico humano.



FONTE: (DATA SCIENCE ACADEMY, 2019b).

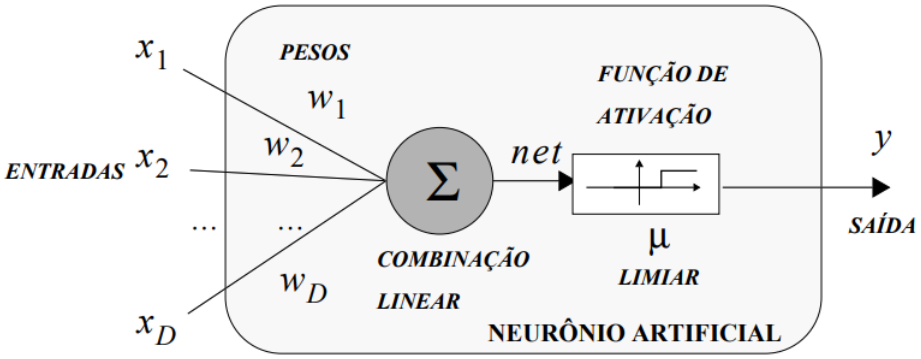
O cérebro é constituído por bilhões de neurônios, havendo entre eles centenas de bilhões de conexões, formando uma enorme rede de comunicação, chamada rede neural. Em cada neurônio há uma corpo celular, diversos dendritos e um axônio. Os dendritos recebem sinais elétricos de outros neurônios por meio das sinapses, que representa o processo de comunicação entre neurônios. O corpo celular faz o processamento dessa informação e envia para outro neurônio. Desse conjunto o corpo celular e os dendritos formam o conjunto de entrada do neurônio e o axônio a saída do fluxo de informação.

A partir da estrutura e funcionamento do neurônio biológico, vários pesquisadores tentaram simular este sistema em computador. Em 1943, os pesquisadores Warren McCulloch e Walter Pitts, apresentaram o modelo mais bem aceito, que de maneira simplificada implementa os componentes e o funcionamento de um neurônio biológico. Nesse modelo matemático, a rede neural artificial é um componente que calcula a soma ponderada de várias entradas, aplica uma função e passa o resultado adiante.

O funcionamento desse modelo relacionado com o neurônio biológico, é descrito da seguinte maneira: Os impulsos elétricos provenientes de outros neurônios são chamados de sinais de entrada, letra x. Dentre os vários estímulos recebidos, alguns excitarão mais e outros menos o neurônio receptor e essa medida é representada através dos pesos sinápticos, definido por wk_n. Onde k representa o índice do neurônio em questão e n se refere ao terminal de entrada da sinapse a qual o peso sináptico se refere. Quanto maior o valor do

peso, mais excitatório é o estímulo. O corpo da célula é representado por dois módulos, o primeiro é o somatório dos estímulos (sinais de entrada) multiplicado pelo seu fator excitatório (pesos sinápticos), e posteriormente uma função de ativação, que definirá com base nas entradas e pesos sinápticos, qual será a saída do neurônio. O axônio representado pela saída yk é obtida pela aplicação da função de ativação. O estímulo pode ser excitatório ou inibitório, representado pelo peso sináptico positivo ou negativo respectivamente, como mostrado na figura 9.

Figura 9 – Visão do Neurônio de McCulloch e Pitts.

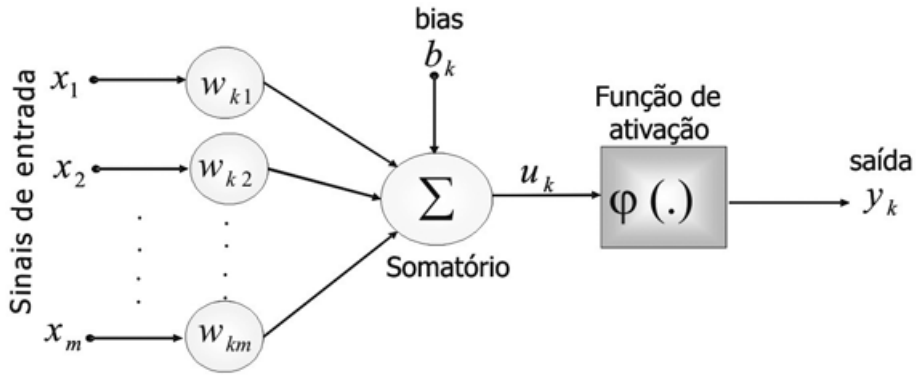


FONTE: (RAUBER, 2005).

O modelo também possui incluída ao somatório da função de ativação uma polarização ou bias de entrada, com o objetivo de aumentar o grau de liberdade desta função e a capacidade de aproximação da rede. Seu valor é ajustado da mesma forma que os pesos sinápticos. Com o bias um neurônio pode apresentar saída não nula, ainda que todas as suas entradas sejam nulas. Por exemplo, caso não houvesse o bias e todas as entradas fossem nulas, então o valor da função de ativação seria nulo. Na figura 10 é mostrado essa descrição.

- **Sinais de entrada (X1, X2, . . . , Xn):** São os dados externos normalmente normalizados que alimentam o modelo preditivo.
- **Pesos sinápticos (W1, W2, . . . , Wn):** São valores usados para ponderar os sinais de entrada da rede. Esses valores são aprendidos durante o treinamento.
- **Combinador linear ($\sum$):** Soma todos os sinais de entrada que foram ponderados pelos respectivos pesos sinápticos, produzindo um potencial de ativação.
- **Limiar de ativação (θ):** Determina qual será o valor escolhido para que o resultado produzido pelo combinador linear possa gerar um valor de disparo de ativação.

Figura 10 – Visão Simplificada do Neurônio Matemático.



FONTE: (VERONEZ et al., 2009).

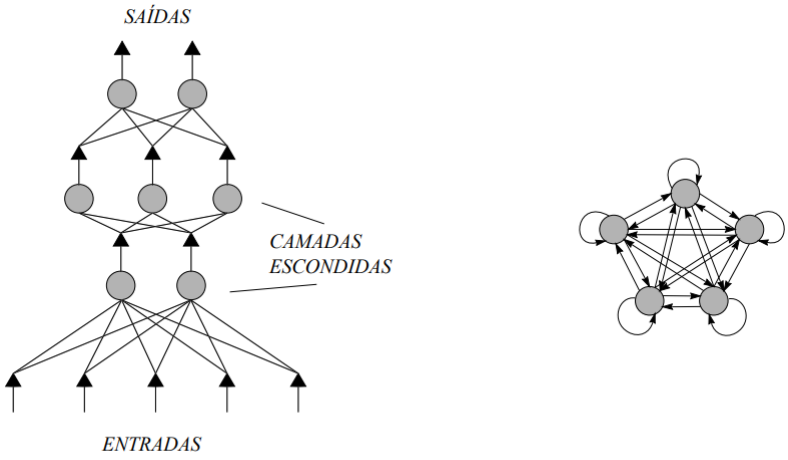
- **Potencial de ativação (u):** É o resultado obtido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. Caso o valor seja positivo (u 0), então o neurônio produz um potencial excitatório. Caso contrário, o potencial será inibitório.
- **Função de ativação (g):** Limita a saída de um neurônio em um intervalo de valores.
- **Sinal de saída (y):** É o valor de saída. Este ainda pode ser usado como entrada de outros neurônios que estão sequencialmente interligados (DATA SCIENCE ACADEMY, 2019b) .

O potencial do cálculo baseado em redes neurais está na criação de conjuntos de neurônios interligados entre si. O processamento local de elementos em paralelo, estabelece a inteligencia da rede. Um elemento da rede recebe um estimulo nas suas entradas, processa esse sinal e origina um novo sinal de saída, que é recebido por outros elementos.

Nesse sentido a fundamentação da topologia dos neurônios pode ser feita em relação ao método de propagação da informação recebia. A diferença é definida em propagação para frente (*feedforward*) e redes realimentadas (*recurrent*). Em redes de propagação para frente o fluxo de informação é unidirecional. Os neurônios que recebem simultaneamente a informação são agrupamentos em camadas. As camadas que não estão conectadas às entradas e nem às saídas da rede são denominadas camadas escondidas ou intermediarias. Exemplos desse tipo de rede são o *perceptron*, o *perceptron* multi-camada e o *ADALINE*.

As redes realimentadas têm conexões entre os neurônios sem restrições. O seu comportamento desempenha um papel fundamental dinâmico. Em certas situações os valores de ativação da rede passam por um processo de relaxação até apresentar um estado estável. Como exemplo se tem a rede auto-associativa. Na figura 11 é mostrada essas topologias (RAUBER, 2005).

Figura 11 – Principais topologias de redes neurais artificiais.



FONTE: (RAUBER, 2005).

2.5.2 Support Vector Machines (SVM)

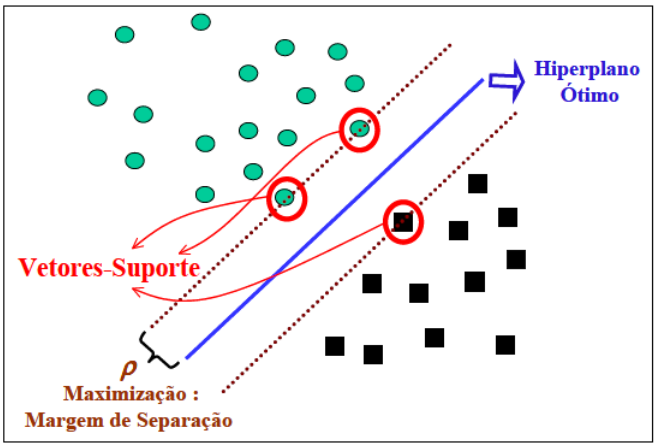
A técnica *SVM*, em português *Máquinas de Vetores de Suporte* é um método de aprendizagem para problemas de reconhecimento de padrões. Foi introduzida por Vapnik (1995) através da teoria estatística de aprendizagem, no qual utiliza do fundamento de separação ótima entre classes, onde se as classes são separáveis, então o resultado é obtido de modo a separar o máximo as classes. (NASCIMENTO et al., 2009).

Sua proposta está em resolver problemas de classificação e análise de regressão. No entanto, usa-se em problemas de classificação de duas classes, isto é, atuando como um classificador binário, onde é aplicado em tarefas de classificação de dados lineares e não lineares (Ahmad et al., 2018).

O funcionamento básico de uma SVM ocorre através do fornecimento de duas classes e um conjunto de pontos que pertencem a essas classes. Com essas informações um hiperplano é determinado separando os pontos de forma a colocar o maior número de pontos da mesma classe do mesmo lado, enquanto atribui o valor mais alto a distância de cada classe a esse hiperplano. A distância de uma classe a um hiperplano é a menor distância entre ele e os pontos dessa classe, no qual é denominado margem de separação. Já a construção do hiperplano é feita por um subconjunto dos pontos das duas classes, chamados de vetor de suporte (GEVERT et al., 2010).

Sendo os dados de treinamento separáveis, o hiperplano ótimo no espaço de características é o que apresenta a máxima margem de separação p, como mostrado na figura 12. Em dados de treinamento onde as amostras das diversas classes não são separáveis (superposição), uma generalização dos dados deve ser realizada (ZUBEN; ATTUX,).

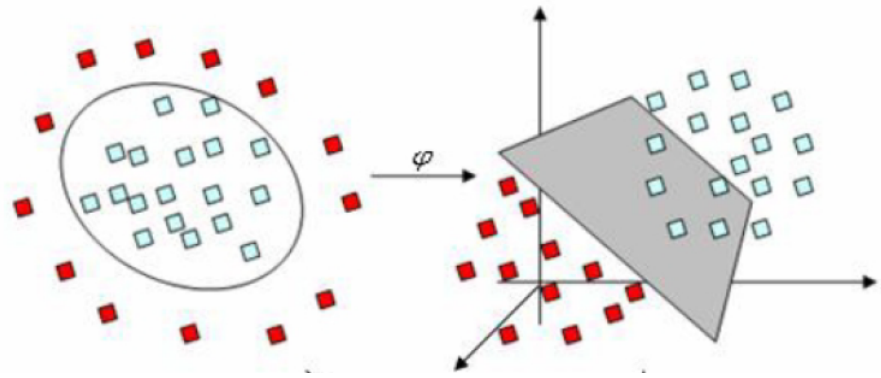
Figura 12 – Dados separados linearmente através do hiperplano que permite maior maximização da margem.



FONTE: (ZUBEN; ATTUX,).

Quando não é possível separar os dados linearmente, é utilizado um classificador não linear que usa funções do kernel para estimar as margens. O objetivo dessas funções é maximizar as margens entre os hiperplanos, sendo as funções do kernel mais empregadas a linear, polinomial, base radial e sigmóide (Ahmad et al., 2018). No funcionamento dessas funções os dados de entrada são mapeados em um espaço de dimensão maior, onde os dados são separados por meio de um hiperplano, tornando-os linearmente separáveis, como mostrado na figura 13 (GEVERT et al., 2010).

Figura 13 – Dados não separáveis linearmente projetados de um plano bidimensional para um plano tridimensional.



FONTE: (GEVERT et al., 2010).

Exemplos de aplicações onde o SVM pode ser encontrado estão em vários contextos como na categorização de textos, na análise de imagens e em Bioinformática (LORENA; CARVALHO, 2007).

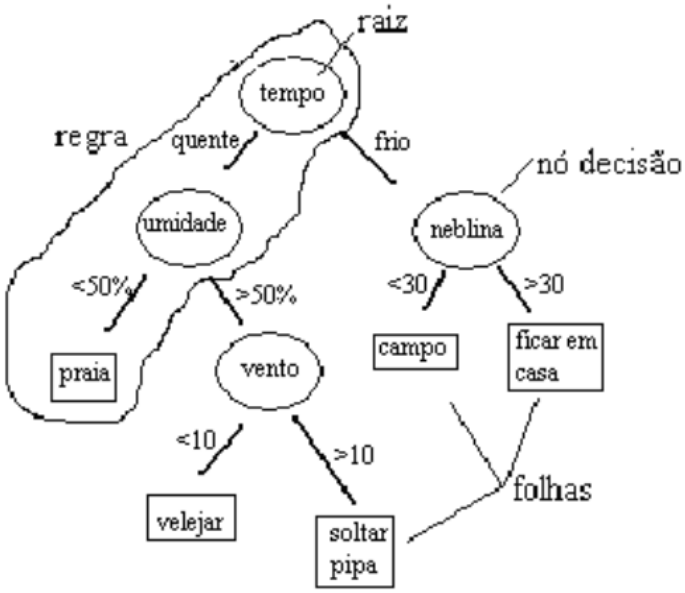
2.5.3 Decision Trees (DT)

A técnica *DT*, em português *Arvores de Decisão*, é um método usado em problemas de classificação e regressão, que particiona recursivamente um conjunto de treinamento, até que cada subconjunto resultante apresente casos de uma única classe. Para isso, na construção da árvore o algoritmo escolhido examina e compara a distribuição de classes.

A construção da árvore de decisão é fundamentada no modelo Top-Down, isto é, a sua estrutura inicia do nó raiz e vai até as suas folhas. Apesar dos algoritmos desse grupo conterem diferenças importantes em sua lógica de execução, todos utilizam da técnica de dividir para conquistar. Esta técnica divide o problema sucessivamente em vário problemas menores, até uma solução para cada um dos problemas mais simples seja encontrada. Nesse fundamento, os algoritmos tentam descobrir maneiras de dividir sucessivamente o conjunto em vários subconjuntos, até que cada um apresente apenas uma classe ou que uma das classes seja a maioria, não sendo necessário novas divisões (CASTANHEIRA, 2008).

Na figura 15 é mostrado um exemplo de uma árvore de decisão. Em sua estrutura os seguintes termos são utilizados:

Figura 14 – Estrutura de uma árvore de decisão.



FONTE: (CASTANHEIRA, 2008).

- **Nó:** São todos os elementos que aparecem na árvore;
- **Nó Raiz:** É o primeiro nó do topo da árvore;

- **Nó pai e nó filho:** São os nós logo abaixo do nó raiz que se dividem em sub nós. Um nó dividido em sub nós é chamado nó pai de sub nós, enquanto os sub nós são filhos do nó pai;
- **Nó de Decisão:** Quando um sub nó se divide em outros sub nós.
- **Folhas ou Terminal:** São os nós que não têm filhos, os últimos elementos da árvore;

Em sua interpretação cada nó de decisão realiza um teste para algum atributo, cada ramo descendente representa um valor desse atributo, o conjunto de ramificações são diferentes, as folhas definem uma classe, e cada percurso da árvore iniciando do nó raiz até a folha, constitui uma regra de classificação. Analisando essa estrutura, é possível extrair regras do tipo "se-então"para melhor compreensão dos resultados.

Para definir as partições da árvore, a regra utilizada é o quão útil o atributo é para a classificação. A partir dessa regra, um determinado ganho de informação é aplicado a cada atributo. Dessa forma, o atributo escolhido como atributo teste para o nó corrente, é o que contém o maior ganho de informação. Por meio dessa aplicação, um novo processo de partição é iniciado.

Um problema encontrado no uso de árvores de decisão é o *Overfitting*, que significa o ajuste exagerado dos dados de treinamento. Na execução do algoritmo de partição recursiva, o mesmo estende a sua profundidade até o ponto de classificar corretamente os elementos do conjunto de treinamento. Nesse processo, quando o conjunto de treinamento não possui ruído, o número de erros no treinamento pode ser zero. Porém, quando o conjunto possui ruído, ou quando o conjunto de treinamento não é representativo, este algoritmo pode produzir o *overfitting*.

Segundo Breiman (1998), uma maneira de impedir o problema do *overfitting* e melhorar a classificação é fazer a poda da árvore. A podagem da árvore tende a ser feita em duas situações. Para parar o crescimento da árvore mais cedo, conhecido como pré-podagem ou poda descendente. Ou com a árvore já completa, conhecido como pós-podagem ou poda ascendente.

Para determinar o número ideal de nós, utiliza-se uma representação gráfica que mostra o percentual de erro no conjunto de treinamento, versus o número de nós da árvore. Quando o erro no conjunto de teste começar a crescer, o número de nós nesse ponto é considerado ideal(SILVA, 2005).

Algoritmos que usam a ideia das árvores de decisão são bastante aplicados por retornarem modelos preditivos de fácil interpretação, estabilidade e alta precisão. O método *Top-Down Induction of Decision Tree* (TDIDT) é um dos mais utilizados na aplicação de árvore de decisão e é referencia para outros algoritmos como o *Classification and Regression Tree* (CART) e o C5.0.

Em sua execução o TDIDT faz uma busca recursiva por atributos que melhor dividem o conjunto de observações em sub-conjuntos. O CART é caracterizado por ter a capacidade de assimilar relações entre dados, mesmo não havendo visíveis relações. Em seu uso faz a construção de árvores binarias, onde cada nó interno possui como saída dois nós filhos. Já no C5.0 o algoritmo transforma árvores treinadas em conjuntos de regras if-then, e avalia qual ordem essas regras devem ser aplicadas (ARRIGONI, 2018)

No seu uso prático, as árvores de decisão foram aplicadas em vários domínios da ciência e engenharia como pesquisas de produtos farmacêuticos, saúde pública, biologia celular, consumo de energia elétrica, estudos de transporte, entre outros(Rivera-Lopez; Canul-Reich, 2018).

2.5.4 Random Forest (RF)

A técnica *RF*, em português *Floresta Aleatória*, é um método desenvolvido por Breiman (2001), para resolver problemas de classificação e regressão de métodos de aprendizagem em árvores, por meio do bootstrap dos dados de treinamento, aumentando o uso do algoritmo conhecido como CART (classification e Regression Trees) proposto por Breiman et al. (1984).

Na grande maioria, os métodos de decisão em árvore possuem viés baixo, mas alta variância, resultando na produção de um sobre ajuste (overfitting). Utilizando RF o problema da variância é tratado usando um número n de árvores de modo que ao produzir n observações independentes f_1, f_2, ..., f_n, uma para cada nó final de árvores, a variância diminui através da média dessa n observações. Isso porque a variância individual de cada árvore é maior do que a variância da média delas (MOTA, 2019).

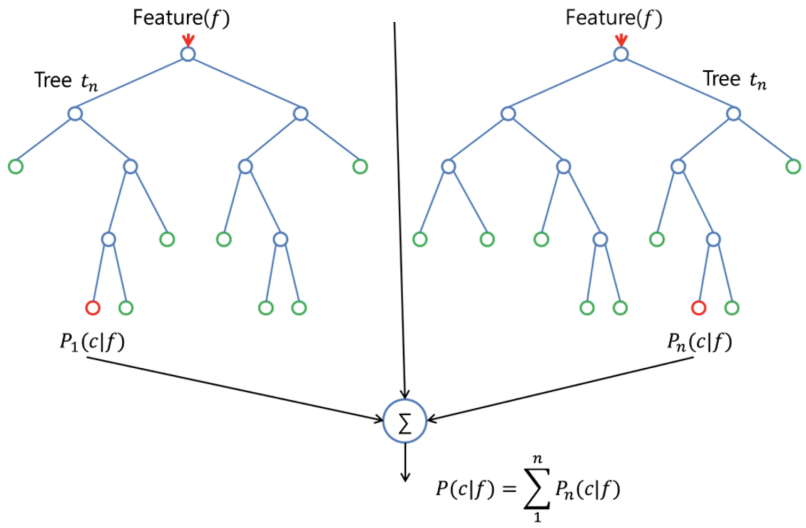
O termo floresta é utilizado, pois a técnica cria uma combinação (*ensemble*) de árvores de decisão, que na maiorias das vezes são treinadas com o método de *bagging*. Este método possui como fundamento a ideia que a combinação dos modelos de aprendizado aumenta o resultado geral. Em uma definição resumida, o algoritmo de *Random Forest* gera várias árvores de decisão e as combinam para conseguir uma predição com maior acurácia e mais estabilidade (MEDIUM, 2018a).

A precisão do classificador de RF é muito boa, possui forte capacidade de generalização, tem alta velocidade computacional e os parâmetros configuráveis são muito poucos. Em particular, o RF é apropriado para calcular a função não linear de variáveis e pode refletir a interação entre variáveis (Liu et al., 2019).

O processo de treinamento do algoritmo RF é baseado no método de bagging. No seu funcionamento, Considerando um conjunto de dados D, com d tuplas. A cada interação i, onde ($i = 1,2,...,k$) um conjunto de treinamento D_i de d tuplas é amostrado com elementos do conjunto de dados D. Isto é, do conjunto de dados iniciais, k amostras são

formadas com instâncias e atributos escolhidos de forma aleatória. Assim, cada conjunto D_i passa pelo processo de treinamento de uma árvore de decisão. Através da média do resultado de todas as árvores o resultado final é obtido (CONTI et al., 2019). Na figura 15 é mostrado uma estrutura genérica do funcionamento desse algoritmo com duas árvores de decisão.

Figura 15 – Estrutura de uma *Random Forest* com duas *Decision Trees*.



FONTE: (MEDIUM, 2018a).

2.5.5 Naive Bayes (NB)

A técnica NB é um modelo que foi proposto pelo reverendo Thomas Bayes no século XVIII(TORRES, 2005). Ela é um algoritmo classificador probabilístico que calcula um conjunto de probabilidades, contando as combinações e a frequência de valores em um determinado conjunto de dados.

O algoritmo utiliza o teorema de Bayes e considera que todos os atributos são independentes, fornecido o valor da variável de classe. O algoritmo demostra aprender rapidamente e ter um bom desempenho em vários problemas de classificação supervisionada.

Para o cálculo das probabilidades é utilizado o teorema de Bayes. Nesse cálculo a probabilidade de um documento d com o vetor x = $<x_1, ..., x_n>$ pertencer à hipótese h é mostrado na equação 2.6:

$$P(h_1|x_i) = \frac{P(x_i|h_1)P(h_1)}{P(x_i|h_1)P(h_1) + P(x_i|h_2)P(h_2)} \tag{2.6}$$

Onde, $P(h_1|x_i)$ é uma probabilidade posterior, e $P(h_1)$ é a probabilidade anterior associada à hipótese h_1. Para m hipóteses diferentes, $P(x_i)$ é descrito pela equação 2.7:

$$P(x_i) = \sum_{j=1}^{n} P(x_i|h_j)P(h_j) \tag{2.7}$$

Dessa forma a equação 2.6 pode ser modificada, como mostrado na equação 2.8 abaixo (PATIL; SHEREKAR et al., 2013).

$$P(h_1|x_i) = \frac{P(x_i|h_1)P(h_1)}{P(x_i)} \tag{2.8}$$

2.5.6 K-Nearest Neighbors (K-NN)

A técnica *K-NN*, em português *K-Vizinhos mais Próximos*, é um método classificador onde o aprendizado é fundamentado na similaridade que um dado (vetor) tem de outro. O treinamento utiliza um conjunto de dados composto por vetores n-dimensionais e cada elemento desse conjunto defini um ponto no espaço n-dimensional.

No seu funcionamento, para descobrir a classe de um elemento que não pertença ao conjunto de dados de treinamento, o KNN busca K elementos do conjunto que estejam mais próximos do elemento desconhecido, isto é, que tenham a menor distância. Estes K elementos são chamados de K-vizinhos mais próximos, e verificando a qual classes esses K vizinhos pertencem, a classe com maior número de elementos será atribuída à classe do elemento desconhecido .

Para o cálculo da distância entre dois pontos, as métricas mais comuns são a distância Euclidiana, Manhattan e Minkowski, sendo a primeira a mais utilizada. A baixo é descrito este cálculo.

Seja $X = (x_1, x_2, ..., x_n)$ e $Y = (y_1, y_2, ..., y_n)$ dois pontos no $\Re^n$

- A distância Euclidiana entre X e Y é descrita pela equação 2.9:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \tag{2.9}$$

Se cada variável for considerada ter certa importância, um peso relativo pode ser incluso, desse modo a distância Euclidiana ponderada pode ser descrita como mostrado na equação 2.10. Nas distâncias Manhattan e Minkowski, os pesos também podem ser aplicados (SILVA, 2005).

$$d(x, y) = \sqrt{w_1(x_1 - y_1)^2 + w_2(x_2 - y_2)^2 + ... + w_n(x_n - y_n)^2} \tag{2.10}$$

Na utilização do algoritmo KNN, a sua execução ocorre por meio das seguintes etapas:

1. Recebe um dado não classificado;
2. Mede a distância (Euclidiana, Manhattan, Minkowski ou Ponderada) do dado desconhecido com todos os outros dados que já estão classificados;
3. Obtém-se as K menores distâncias;
4. Verifica-se a classe de cada um dos dados que tiveram a menor distância e se conta a quantidade de cada classe que aparece;
5. Se escolhe como resultado a classe que mais apareceu dentre os dados que tiveram as menores distâncias;
6. Classifica-se o dado desconhecido com a classe escolhida do resultado da classificação que mais apareceu.

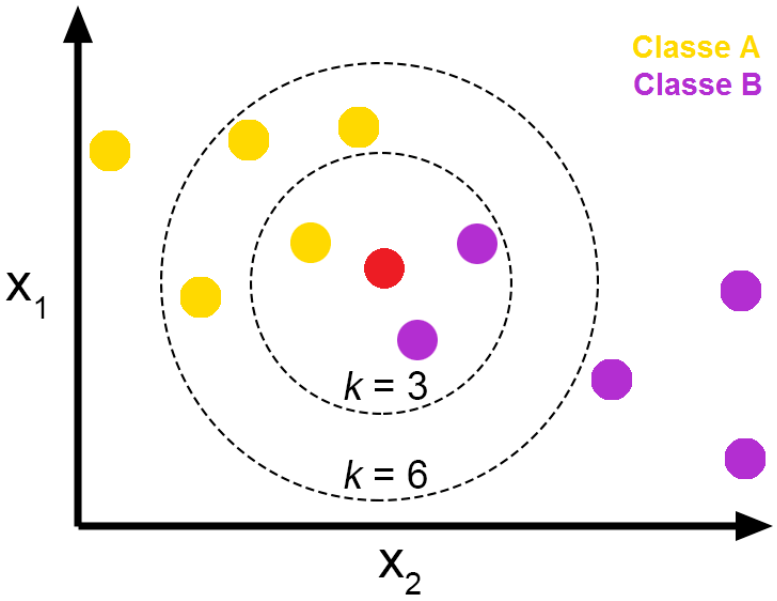
Na figura 16 é mostrado esse processo. Pode-se observar um dado não classificado representado em vermelho, e todos os os outros dados já classificados, definidos em amarelo e roxo, que respectivamente representam a classe A e a classe B. O cálculo da distância do dado desconhecido com todos os outros é feito para verificar quais estão mais próximos, isto é, tem as menores distâncias. Com o resultado do cálculo, se escolhe três ou seis dos dados mais próximos e verifica qual é a classe que mais aparece. Nesse exemplo é usado K igual a três. Os dados mais próximos do dado desconhecido são os que estão dentro do primeiro círculo (olhando de dentro para fora). Nele há três dados já classificados, e observando a classe que mais predomina, se conclui que o dado roxo (Classe B) é o mais predominante, pois possui dois dados em relação ao amarelo que possui só um. Desse modo, o dado desconhecido passa a ser classificado como sendo o roxo da classe B. Caso o valor de K seja igual a 6, o dado desconhecido passa ser classificado como sendo o amarelo da classe A (MEDIUM, 2018b).

O classificador KNN possui o número de K-vizinhos como único parâmetro livre, o qual é controlado pelo usuário para obter uma melhor classificação. Esta classificação pode ser exaustiva computacionalmente se utilizado um conjunto com muitos dados. Entretanto, em aplicações específicas seu uso é recomendado (SILVA, 2005).

2.6 Validação de algoritmos de aprendizado de máquina

Após realizar a criação e treinamento de um modelo de aprendizado de máquina **AM!** (**AM!**), o principal objetivo é escolher o modelo que faça as melhores previsões, o que

Figura 16 – Classificação de um dado desconhecido utilizando k-NN.



FONTE: (MEDIUM, 2018b).

representa selecionar o modelo que teve a melhor configuração no treinamento. Entretanto, ao selecionar esse modelo, como o mesmo foi treinado com os dados de treinamento, isso pode causar sobreajuste (overfitting) no modelo.

2.6.1 Sobreajuste (Overfitting)

O sobreajuste acontece quando um modelo aprende padrões vistos nos dados de treino, desempenhando um resultado satisfatório, mas não consegue generalizar os padrões em dados não vistos. Geralmente isso ocorre quando os dados de treinamento incluem todos os dados usados na avaliação do modelo, desse modo o modelo memoriza os dados reconhecidos, mas não consegue fazer previsões certas com dados novos.

Para tentar prevenir essa situação, pode-se reservar dados adicionais para validar o desempenho do modelo. Como exemplo, pode-se dividir o conjunto de dados em 60% para treinamento, 20% para teste e 20% para validação. Assim, feito o treinamento com a escolha dos melhores parâmetros do modelo, que apresentaram resultados satisfatórios com os dados de teste. Se realiza uma nova avaliação do modelo com os dados de validação. Se os resultados forem semelhantes aos dados de teste, então o modelo não está apresentando sobreajuste (AMAZON, 2016).

Ao usar um terceiro conjunto de dados para validar o modelo treinado, isso ajuda a escolher os melhores parâmetros do modelo de treinamento e com isso evitar o sobreajuste. Porém, dividir o conjunto de dados em dados de teste e validação, resulta em menos dados para o treinamento do modelo. O que para um conjunto de dados pequeno, é um problema,

pois o ideal é utilizar o máximo possível de dados no treinamento. Desse modo como alternativa a essa situação, pode-se usar a validação cruzada (Cross Validation)

2.6.2 Validação Cruzada (Cross Validation)

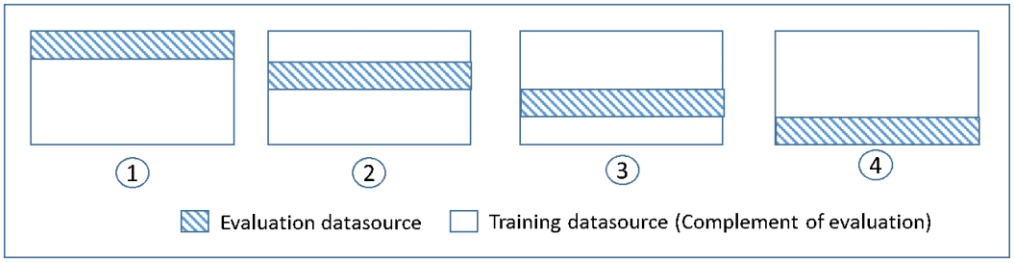
A validação cruzada é uma técnica usada para avaliar modelos de aprendizado de maquina através do treinamento do modelo por meio de subconjuntos de dados de entrada disponíveis representando os dados de treino e a sua avaliação por meio de um subconjunto complementar dos dados representando os dados de teste. Com o objetivo de detectar se o modelo treinado apresenta sobreajuste (Overfitting). Para a utilização da validação cruzada no modelo um método muito utilizado é a validação cruzada *k-fold*.

2.6.2.1 Validação cruzada k-fold (Cross Validation k-fold)

A validação cruzada *k-fold* é um método que divide o conjunto de dados de entrada em subconjuntos de dados k denominados *folds*. Dessa divisão, se utiliza *k-1* dos subconjuntos para treinar o modelo e o subconjunto restante que não foi usado para o treino, é utilizado para testa-lo. Esse processo é então repetido k vezes, com um subconjunto de dados diferente reservado para teste e o restante para treinamento do modelo a cada vez, até que todos os subconjuntos tenham sido usados para testar o modelo. Para determinar o resultado final de desempenho do modelo, é feita a média dos resultados obtidos em cada repetição (AMAZON, 2016).

Para exemplificar esse processo, na figura 17 é mostrado um exemplo da validação cruzada k-fold com k igual a 4. Com esse valor o conjunto de dados selecionado que representa os dados de treino vai ser dividido em 4 subconjuntos, onde também vai ocorrer 4 repetições de treino e teste. Na primeira repetição, chamada de modelo 1, 75% dos subconjuntos são usados para treino e os 25% restantes para teste, isto é, a primeira divisão do subconjunto é usada para teste e as demais para treino. No modelo 2, é usado o segundo subconjunto para teste e os demais para treino, e assim por diante até realizar os teste com todos os subconjuntos.

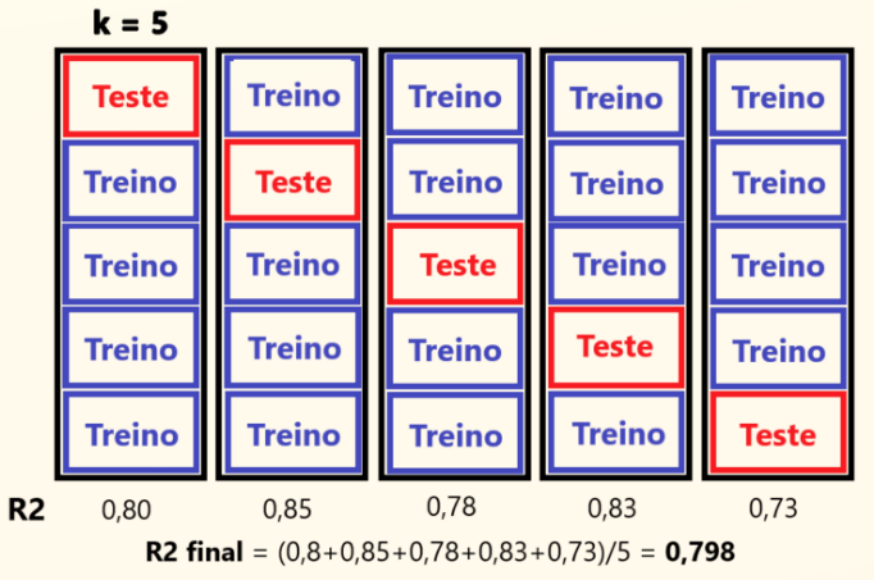
Figura 17 – Exemplificação de validação cruzada k-fold com k igual a 4



FONTE: (AMAZON, 2016).

Terminadas todas as repetições de treino e teste, ao final são gerados 4 resultados avaliativos de desempenho, uma para cada modelo. Para obter o resultado do desempenho geral é realizado a média dos quatro resultados. Na figura 18 é mostrado outro exemplo com o valor de k igual a 5, mostrando também o exemplo da média dos resultados.

Figura 18 – Exemplificação de validação cruzada k-fold com k igual a 5



FONTE: (MEDIUM, 2018b).

Obtido o resultado, o desempenho do modelo pode ser avaliado. No entanto, esse resultado é conseguido com os dados de treino que são dados que o modelo já conhece. Para verificar realmente o desempenho, deve-se usar os dados de teste por meio da técnica *Matriz de Confusão*.

2.6.3 Matriz de Confusão (Confusion Matrix)

Uma matriz de confusão é uma tabela que categorizará as previsões em relação aos valores reais para cada classe do modelo. Ela possui duas dimensões sendo que cada linha na matriz representará os valores reais e as colunas serão responsáveis pelos valores previstos de cada classe do modelo. Entretanto isso também pode ser o contrário, tendo que se verificar qual é a referência no método utilizado. O termo matriz de confusão é usado pois mesmo que a matriz seja fácil de analisar, os termos usados na sua análise podem parecer um pouco complexo inicialmente, gerando uma certa confusão. No seu uso, pode-se criar uma matriz de qualquer número de valores de classe, sendo que o valor mínimo da matriz deve ser 2x2. Na tabela 2 é mostrado como exemplo a matriz de confusão de duas classes junto com a indicação da distribuição dos valores previstos e reais. Em sua interpretação de avaliação do desempenho, a matriz vai escolher uma classe como referência sendo denominada como classe positiva e a outra classe do conjunto como a

negativa (CN, 2022). A partir disso ela vai categorizar as classes em termos de ocorrência de previsões positivas ou negativas como descrito abaixo:

Tabela 2 – Matriz de confusão de 2 classes

		Classes de referência		**Total previsto**
		P	**N**	
Classes previstas	**P**	VP	FP	$\sum_l P$
	N	FN	VN	$\sum_l N$
Total referência		$\sum_c P$	$\sum_c N$	$\sum T$

FONTE: Elaborada pelo autor.

- **Verdadeiro Positivo - VP (True Positive — TP):** Representa que no conjunto real, a classe que se está prevendo foi prevista corretamente. **Exemplo:** Quando a voz verificada é feminina e o modelo previu corretamente que a voz é feminina.
- **Falso Positivo - FP (False Positive — FP):** Representa que no conjunto real, a classe que se está prevendo foi prevista incorretamente. **Exemplo:** Quando a voz verificada é feminina e o modelo previu incorretamente que a voz não é feminina, isto é, é masculina.
- **Falso Negativo - FN (False Negative — FN):** Representa que no conjunto real a classe que não se está prevendo foi prevista incorretamente. **Exemplo:** Quando a voz verificada não é a feminina, no caso a masculina, e o modelo previu incorretamente que a voz é feminina.
- **Verdadeiro Negativo - VN (True Negative — TN):** Representa que no conjunto real, a classe que não se está prevendo foi prevista corretamente. **Exemplo:** Quando a voz verificada não é a feminina, no caso a masculina, e o modelo previu corretamente que a voz não é feminina.

2.6.3.1 Métricas de Avaliação da matriz de confusão

Ao se utilizar a matriz de confusão para se contar o acerto dos dados previstos em relação aos resultados dos dados de teste, ela fornece através desses resultados o cálculo de métricas que avaliam o desempenho da classificação do modelo, sendo elas:

- **Acurácia (accuracy):** Fornece a performance geral de previsões corretas do modelo. Ou seja, de todas as classificações, quantas o modelo classificou corretamente, como mostrado na equação 2.11.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.11}$$

- **Taxa de erro:** Fornece a performance geral de previsões erradas do modelo. Ou seja, de todas as classificações, quantas o modelo classificou errado, como mostrado na equação 2.12.

$$Taxa\ de\ erro = 1 - Acurácia \tag{2.12}$$

- **Sensibilidade/Recall/Revocação:** Fornece a performance do acerto de classe positiva esperada. Ou seja, de todas as situações de classe Positiva como valor esperado, quantas estão certas, como mostrado na equação 2.13;

$$Sensibilidade = \frac{VP}{VP + FN} \tag{2.13}$$

- **Especificidade:** Fornece a performance de acerto de classe negativa prevista. Isto é, de todas as classificações de classe Negativa que o modelo fez, quantas ele acertou, como mostrado na equação 2.14.

$$Especificidade = \frac{VN}{VN + FP} \tag{2.14}$$

- **Eficiência:** Fornece a média aritmética da Sensibilidade e Especificidade, como mostrado na equação 2.15 (MARIANO, 2021).

$$Eficiência = \frac{Sensibilidade + Especificidade}{2} \tag{2.15}$$

3 DESENVOLVIMENTO

3.1 Base de dados

No desenvolvimento deste trabalho foi usado uma base de dados chamada *Gender Recognition by Voice* adquirida no site *data.world* no formato de um arquivo *.csv* pelo link *https://data.world/ml-research/gender-recognition-by-voice*. Essa base é constituída com milhares de amostras de vozes femininas e masculinas, cada uma estando rotulada pelo seu gênero. Segundo o site, ela é composta por amostras de vozes que foram coletadas de quatro fontes distintas, sendo elas:

- The Harvard-Haskins Database of Regularly-Timed Speech (O banco de dados Harvard-Haskins de fala em tempo regular)
- Telecommunications Signal Processing Laboratory (TSP) Speech Database at McGill University (Banco de dados de fala do Laboratório de processamento de sinais e telecomunicações (TSP) da Universidade McGill)
- VoxForge Speech Corpus
- Festvox CMU_ARCTIC Speech Database at Carnegie Mellon University (Festvox CMU_ARCTIC do banco de dados de fala da universidade de Carnegie Mellon)

Além disso, o mesmo informa que para a geração das informações da base de dados, cada amostra de voz foi armazenada como um arquivo *.wav*, que então foi pré-processado por análise acústica utilizando a função *Specan* do pacote *WarbleR* da linguagem R. Essa função mede 22 parâmetros acústicos em sinais para os quais os tempos de início e término do áudio são fornecidos. Na tabela 3 são mostradas as 22 propriedades acústicas resultantes do pré-processamento e a sua descrição.

O resultado do pré-processamento dos arquivos *.wav* foi então salvo em um arquivo *.csv*, possuindo 3.168 linhas e 21 colunas. Sendo que das 21 colunas, as 20 primeiras representam as propriedades acústicas medidas e a última o rótulo (label) para definir a classificação dos gêneros feminino e masculino em relação a essas propriedades, como mostrado na figura 19.

Observando a figura 19, pode-se notar que ela não possui as 22 propriedades acústicas resultantes mencionadas. Sobre isso o site informa que não foram usados as propriedade *duration* e *peakf*, pois a *duration* se refere a duração de gravação, que para o treinamento é interrompida em 20 segundos. E a *peakf* devido a restrições de tempo e

Tabela 3 – Propriedades acústicas resultantes do pré-processamento da função *Specan*

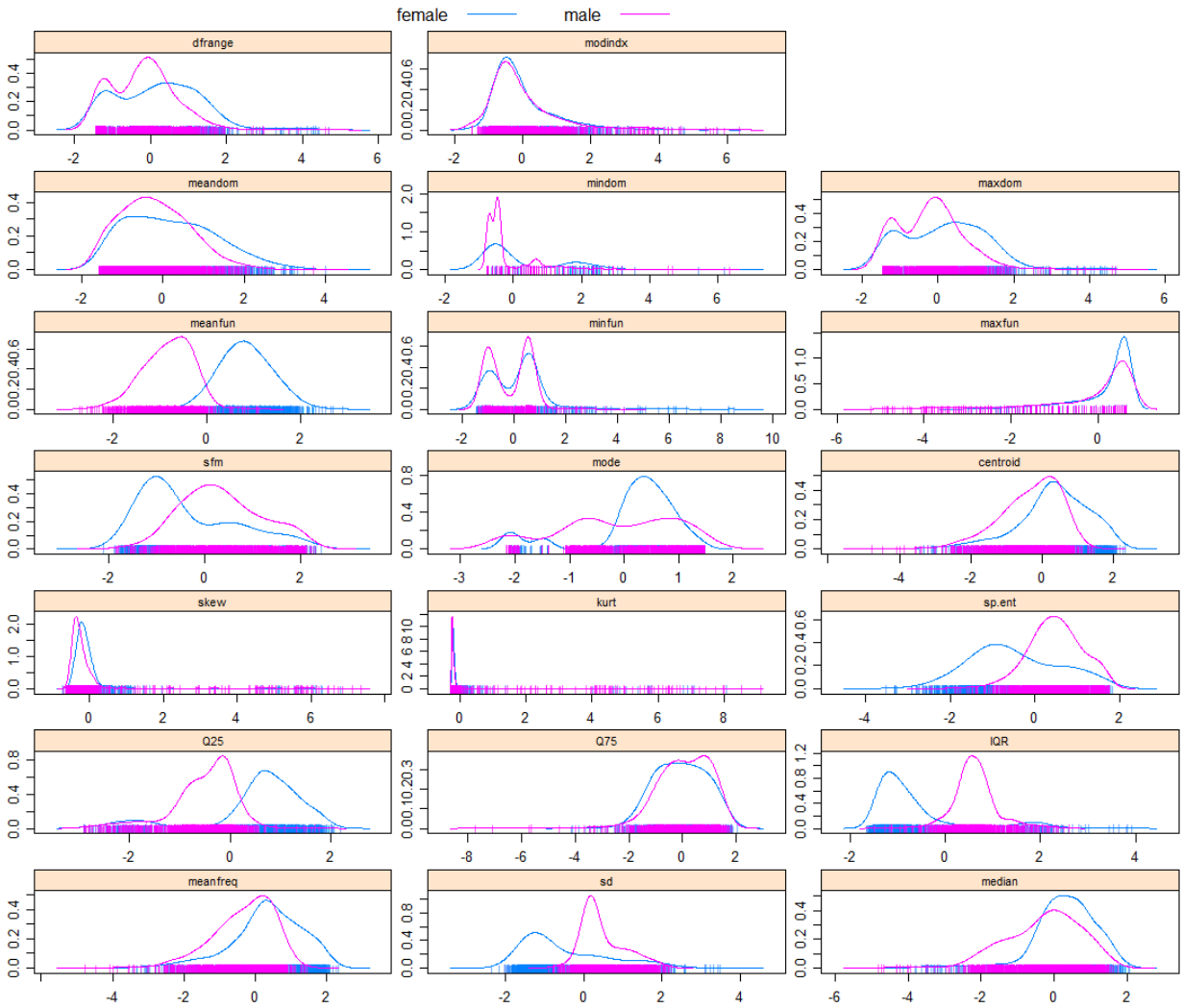
Parâmetro	Descrição
duration	Comprimento do sinal
meanfreq	Frequência média (em kHz)
sd	Desvio padrão da frequência
median	Frequência mediana (em kHz)
Q25	Primeiro quantil (em kHz)
Q75	Terceiro quantil (em kHz)
IQR	Intervalo interquantil (em kHz)
skew	Skewness - Assimetria - Medida da falta de simetria de uma determinada distribuição de frequência
kurt	Kurtosis - Curtose - Uma medida de pico
sp.ent	Entropia espectral
sfm	Planura espectral
mode	Modo de frequência
centroid	Centróide de frequência (ver specprop)
peakf	Frequência de pico (frequência com maior energia)
meanfun	Média da frequência fundamental medida através do sinal acústico
minfun	Frequência fundamental mínima medida através do sinal acústico
maxfun	Frequência fundamental máxima medida através do sinal acústico
meandom	Média da frequência dominante medida através do sinal acústico
mindom	Mínimo de frequência dominante medida através do sinal acústico
maxdom	Máximo da frequência dominante medida através do sinal acústico
dfrange	Faixa de frequência dominante medida através do sinal acústico
modindx	Índice de modulação. Calculado como a diferença absoluta acumulada entre medições adjacentes de frequências fundamentais dividida pela faixa de frequência

FONTE: Elaborada pelo autor.

CPU no cálculo do valor. Desse modo, caso essas informações fossem usadas, então todas as amostras teriam o mesmo valor para *duration* sendo igual a 20 e *peakf* igual a 0. Mas como esses valores ficaram constantes para todas as amostras, eles foram desconsiderados.

Na figura 19, além de mostrar as características extraídas das amostras de vozes dos arquivos *.wav*, ela também mostra as funções de densidade de probabilidade *Probability Density Function (PDF)* dos tipos de gênero feminino e masculino em relação a cada propriedade. Com o seu uso, é possível fazer uma identificação para prever quais das propriedades das amostras de dados é mais significativa para a classificação dos gêneros. Para isso é preciso que a distribuição de cada rótulo resultante da função seja diferente no eixo vertical e no eixo horizontal. Isto é, ao analisar por exemplo, as propriedades *IQR* e *meanfun* pode-se notar que a distribuição de voz feminina (female) e masculina(male) estão apresentadas em regiões diferentes no eixo das abcissas (eixo x), representando que essas propriedades são ótimas candidatas para diferenciar os rótulos em análise. No entanto, olhando para as propriedades *modindx* e *kurt*, as distribuições estão quase sobreposta uma sobre a outra, não havendo uma certa diferença, o que determina que essas propriedades não seriam muio indicadas para reconhecer os gêneros.

Figura 19 – Densidade de probabilidade das características presentes no conjunto de dados.



FONTE: Elaborada pelo autor.

Contudo, mesmo as propriedades *modindx* e *kurt* apresentando a sua distribuição não muito diferenciada em relação as demais, se decidiu utilizar todas as propriedades do conjunto de dados para a análise, para ver o resultado com essa configuração.

3.2 Preparação da base de dados

Com a definição das propriedades a serem usadas da base de dados, antes que a mesma seja utilizada no algoritmo de treinamento, é indicado que se verifique os seus dados, e veja se necessitam de algum tratamento, isto é, transforma-los em outros valores ou retirá-los se necessário. Para a base de dados utilizada, isso não foi preciso, pois na sua verificação se constatou que os dados já estavam prontos para uso. No site onde a mesma foi adquirida, não foi descrito se algum tratamento foi feito, assim, se considerou que os

dados já ficaram prontos após o uso da função *Specan*.

Estando os dados preparados, para usá-los com os algoritmos das técnicas de aprendizado de máquina, os mesmos foram divididos em duas partes, uma denominada dados de treino, para ser usado no treinamento dos modelos e a outra de dados de teste, para testar a previsão dos modelos treinados.

Para a escolha dos valores em cada conjunto de dados, se usou como referência o que foi apresentado na subseção 2.6.1, em que dois métodos de divisão podem ser utilizados, no qual, no primeiro se escolhe fazer a divisão em 60% para os dados de treino, 20% para os dados de teste e se criaria um terceiro conjunto denominado conjunto de validação possuindo os 20% restantes, para verificar se o modelo não teve sobreajuste (overfitting). No entanto, como esse método cria um terceiro conjunto, caso o *dataset* possua um número pequeno de amostras, isso representa menos dados para o treinamento dos modelos, o que não é algo muito desejado. No segundo método, é indicado usar os conjuntos de treino e teste, colocando-se mais amostras nos dados de treino e no mesmo usar a técnica validação cruzada, com o uso do método validação cruzada *k-fold*, que já faz a verificação de sobreajuste (overfitting) nos modelos treinados.

Como a base de dados escolhida para este trabalho possui um valor considerado de amostras, poderia-se escolher o primeiro método de divisão, contudo, se priorizou usar um número maior de amostras, para se ter mais dados para o aprendizado. Desse modo, se escolheu o método com o uso da validação cruzada e com isso os dados foram divididos em 80% para treino e 20% para teste. Na tabela 4 é mostrado como ficou a divisão das amostras da base de dados, junto com o número de amostras por gênero na divisão de cada conjunto .

Tabela 4 – Divisão da base de dados com número de observações de gênero por conjunto

Dados	**Female**	**Male**	**Total**
Treino (80%)	1268	1268	2536
Teste (20%	316	316	632
Total (100%)	1584	1584	3168

FONTE: Elaborada pelo autor.

3.3 Software escolhido para a implementação

Determinado como usar a base de dados, para a realização do treinamento e avaliação dos resultados das técnicas de aprendizado de máquina, se escolheu para implementar e aplicar esses objetivos o software RStudio (RSTUDIO, 2021). Esse software utiliza a linguagem R para o desenvolvimento das aplicações e foi escolhido por possuir uma fácil manipulação dos conjuntos de dados, além de oferecer vários pacotes que podem ser

importados para uso nos projetos, permitindo realizar novas possibilidades com os dados. Entre esses pacotes estão incluso os destinados para aplicações de técnicas de aprendizado de máquina, o que desse modo facilita na implementação das técnicas escolhidas.

3.3.1 Hardware utilizado para a implementação

Para a implementação do algoritmo de treinamento com o software RStudio se usou um notebook com a configuração de hardware mostrado na tabela 5.

Tabela 5 – Hardware usado para desenvolver o algoritmo de treinamento

Notebook	Dell Inspiron 3583
Processador	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz x 8
Memoria RAM	8 GB
Sistema Operacional	Ubuntu 20.04.2 LTS 64-bit
Software RStudio	2021.09.0 Build 351 64-bit
Software R	R x64 4.1.1

FONTE: Elaborada pelo autor.

3.4 Desenvolvimento do algoritmo de treinamento

Nessa seção é apresentado de um modo geral, como foi feito a implementação do algoritmo para o treinamento dos modelos das técnicas selecionadas usando o software RStudio. Para ver o código completo do algoritmo, além de poder executá-lo e testa-lo, o link do repositório do mesmo é fornecido no apêndice A situado no final deste trabalho.

3.4.1 Bibliotecas usadas

No desenvolvimento do algoritmo se usou como principal recurso o pacote *Classification And Regression Training (caret)*, abreviação de treinamento de classificação e regressão da linguagem R, por oferecer várias funções destinadas no desenvolvimento de projetos de aprendizado de máquina.

3.4.1.1 Pacote *caret*

Esse pacote foi desenvolvido para treinamento de modelos de classificação e regressão, fornecendo um conjunto de funções que simplificam o processo de criação de modelos preditivos, resultando em uma maior agilidade na construção do modelo e no processo de avaliação dos resultados. Dentre essas funções, estão disponíveis uma coleção de ferramentas para divisão dos dados, pré-processamento, seleção de recursos, ajuste de modelo usando re-amostragem, estimativa de importância variável, entre outras funcionalidades (KUHN, 2019a).

Das funções fornecidas se usou três consideradas essenciais, sendo elas, *createDataPartition* para dividir o dataset no conjunto dos dados de treino e dos dados de teste, *train* que representa uma das principais funções desse pacote, destinada para a criação de modelos de aprendizado de máquina e *confusionMatrix* para avaliar as predições dos modelos treinados.

3.4.1.2 Função *createDataPartition*

A função createDataPartition do pacote caret é usada para criar divisões balanceadas dos dados. Dessas divisões o conjunto de dados de treino e de teste são gerados (RDOCUMENTATION, 2022b). Na sua utilização se tem disponíveis 5 parâmetros, nos quais com o uso de três deles já se pode executar a função, como mostrado no código 3.1:

```
1 library(caret)
2
3 set.seed(3456)
4 trainIndex <- createDataPartition(dataset$label, p = 0.8, list = FALSE)
5
6 trainingData <- dataset[ trainIndex,]
7 testData  <- dataset[-trainIndex,]
```

Código 3.1 – Estrutura de uso da funçãocreateDataPartition

Para que os conjuntos de dados gerados das divisões possam ser reproduzidos, deve-se definir uma semente através da função nativa da linguagem R *set.seed(nmeroDaSemente)*. Nos três parâmetros usados, o primeiro serve para escolher uma das colunas do dataset para saber quantas linhas ele têm, e servir de base para o cálculo da divisão, geralmente se utiliza a coluna *label*. O segundo, define a porcentagem que a divisão das amostras deve ser feita, no exemplo está 0.8 que representa 80%. E o terceiro define que o resultado gerado não seja uma lista, que aceita mais de um tipo de dado. Sendo *False*, retorna um vetor do tipo *numeric* que é o desejado nesse contexto.

Com os parâmetros escolhidos, a função é executada e um vetor de índices das linhas é gerado. Com esses índices, os mesmo são informados na escolhas de índices do dataset para selecionar os dados de treino. Para se ter os dados de teste, se usa novamente os índices na escolha de índices do dataset, porém se coloca um sinal de menos na frente da variável dos índices para especificar que deve-se selecionar os índices do datset que não estão contidos nesse vetor. Através desse processo os conjuntos de dados de treino e de teste são criados.

3.4.1.3 Função *train*

Essa função tem como objetivo, criar e treinar modelos de classificação e regressão, através da configuração de uma grade de parâmetros de ajuste. Por meio dessas configura-

ções, o modelo selecionado é ajustado e seu desempenho é calculado usando re-amostragem (RDOCUMENTATION, 2022c). No seu uso, inicialmente deve-se escolher um modelo específico, se tendo 238 modelos diferentes disponíveis entre modelos de classificação ou regressão, e definir os parâmetros que o algoritmo deve avaliar. Definido os parâmetros, o passo seguinte está em especificar o tipo de re-amostragem utilizada. Neste ponto a função oferece métodos de validação cruzada *k-fold* (uma vez ou com repetições), validação cruzada *leave-one-out* e *bootstrap* (estimação simples ou regra 632). Realizado essas configurações e iniciado o treinamento, após o processo de re-amostragem, o modelo ajustado(treinado) apresenta como resultado várias medidas de desempenho para orientar na escolha do modelo ideal com base nos parâmetros de ajuste definidos. Por padrão, a função seleciona os parâmetros relacionados ao melhor valor da medida de desempenho apresentada, podendo-se alterar essa medida de avaliação ou definir outros algoritmos para a escolha dos melhores parâmetros. Terminado a execução desse processo, um modelo final usando os dados de treino é ajustado, utilizando os parâmetros considerados "ótimos"escolhidos (KUHN, 2019b). No código 3.2 é mostrado a lógica de funcionamento da função *train*.

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold-out specific samples
        [Optional] Pre-process the data
        Fit the model on the remainder
        Predict the hold-out samples
    end
    Calculate the average performance across hold-out predictions
end
Determine the optional parameter set
Fit the final model to all training data using the optional parameter set
```

Código 3.2 – Lógica do algoritmo da função *train*

3.4.1.4 Função *confusionMatrix*

Essa função calcula uma tabulação cruzada de classes observadas e previstas com estatísticas associadas (RDOCUMENTATION, 2022a). No seu uso deve-se informar dois parâmetros, o primeiro o resultado da predição do modelo treinado com os dados de teste e no segundo o label dos dados de teste, como mostrado no código 3.3.

```
confusionMatrix(data = test_set$pred, reference = test_set$obs)

# Pode-se declarar assim também
confusionMatrix(test_set$pred, test_set$obs)
```

Código 3.3 – Uso da função *confusionMatrix*

Após a execução da função se tem como resultado as métricas que avaliam o desempenho do modelo, como mostrado no exemplo do código 3.4.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Class1 Class2
##     Class1    183    141
##     Class2     13    663
##
##                Accuracy : 0.846
##                  95% CI : (0.8221, 0.8678)
##     No Information Rate : 0.804
##     P-Value [Acc > NIR] : 0.0003424
##
##                   Kappa : 0.6081
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9337
##             Specificity : 0.8246
##          Pos Pred Value : 0.5648
##          Neg Pred Value : 0.9808
##              Prevalence : 0.1960
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.3240
##       Balanced Accuracy : 0.8792
##
##        'Positive' Class : Class1
```

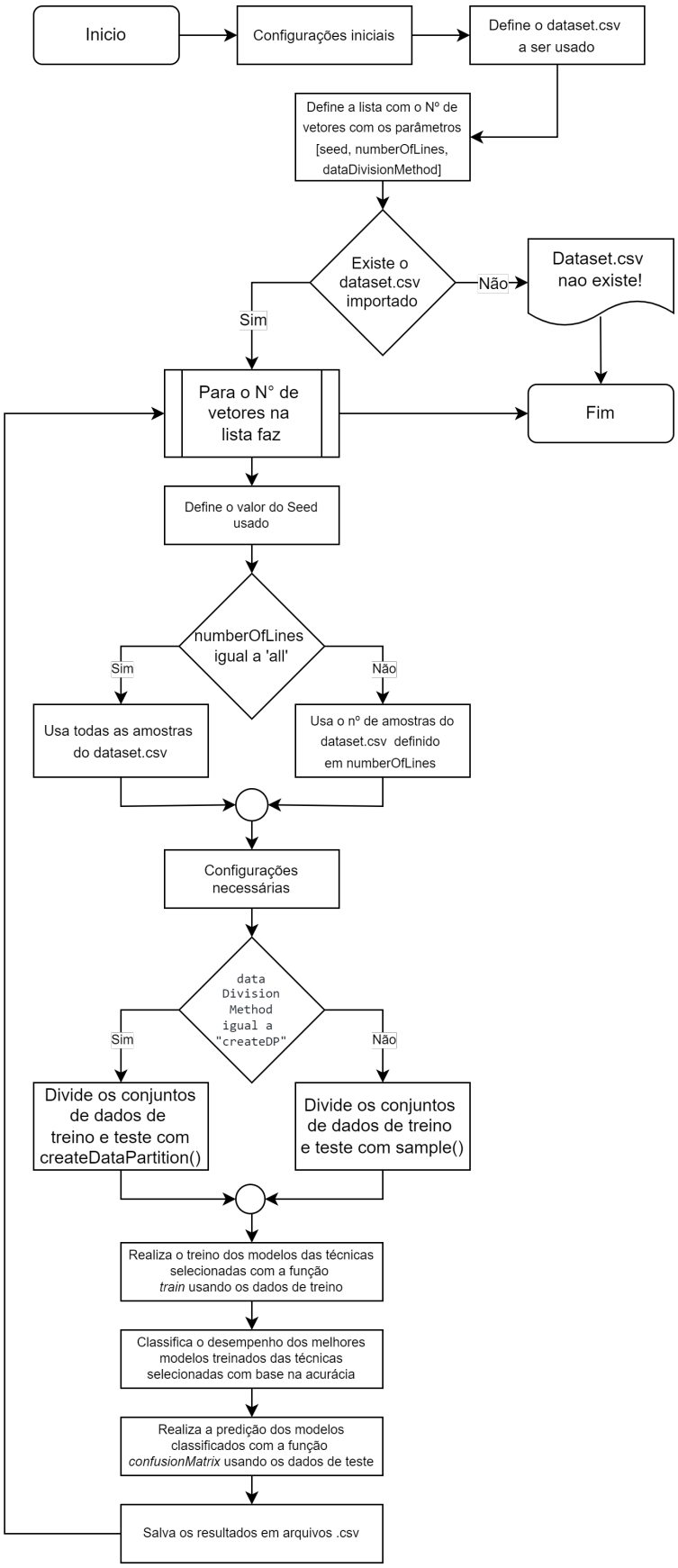
Código 3.4 – Exemplo de retorno da função *confusionMatrix*

3.4.2 Lógica e fluxograma do algoritmo de treinamento desenvolvido

Com as escolhas das funções do pacote *caret* a serem usadas, o algoritmo de treinamento foi criado possuindo de modo geral a seguinte lógica como mostrado no fluxograma da figura 20.

No seu uso, antes de executá-lo deve-se definir no código qual o *dataset* deve-se usar e também foi criado alguns parâmetros que devem ser passados para configurar a análise do treinamento. Para a configuração desses parâmetros, foi determinado usar uma lista de vetores, e nesses vetores devem-se informar os parâmetros da configuração desejada. Através do número de vetores passados, pode-se realizar mais de uma análise de treinamento, se alterando os parâmetros pretendidos. isto é, caso seja definido um vetor, isso significa que será feita uma análise com as informações passadas. Contudo, caso a

Figura 20 – Fluxograma do algoritmo de treinamento



FONTE: Elaborada pelo autor.

lista possua dois ou mais vetores, então será feito a análise de aprendizado de máquina correspondente ao número de vetores passado na lista.

Para configurar os vetores nessa lista, foi especificado três parâmetros sendo eles, *seed, numberOfLines e dataDivisionMethod*. A variável *seed* foi usada para definir uma semente para poder se reproduzir uma análise de treino por outra pessoa. Já a variável *numberOfLines* foi usada para se escolher o número de amostras usadas para a análise. Dessa forma, para escolher usar todas as amostras do *dataset*, foi determinado colocar o valor '*all*'. Caso se queira usar um valor de amostras menor que o total do *dataset*, deve-se colocar um valor numérico menor que esse. Ou seja, o *dataset* selecionado para a análise tem 3168 amostras, caso se queira usar todas as amostras, deve-se colocar o valor '*all*', mas caso se queira menos que isso, deve-se colocar um valo menor, como por exemplo 2000, 1000 ou outro valores abaixo desses. Por último, a variável *dataDivisionMethod* foi determinada para escolher o método usado para dividir os dados de treino e os dados de teste. Nessa escolha, foi definido dois métodos selecionados pelos valores predefinidos '*createDP*' e '*sample-prob*', nos quais, fazem a divisão das amostras em 80% para os dados de treino e 20% para os dados de teste. O valor '*createDP*' representa que a divisão será feita usando a função *createDataPartition()* do pacote *caret*. Já o valor '*sample-prob*' determina que a divisão será feita usando a função *sample()* nativa da linguagem R. O método escolhido por padrão para a análise deste trabalho foi o '*createDP*' pois divide os dados de treino e teste igualmente. A opção *sample()* foi inserida no algoritmo, pois ela foi usada nos testes iniciais para a divisão das amostras, mas se notou que ela não divide igualmente os dados para os dados de treino e teste, assim esse método foi deixado como opção para análises futuras, para ver se o resultado de treino usando esse método resulta no mesmo resultado ou em valores diferentes. No código 3.5 é mostrado um exemplo de como foi feita a configuração dos parâmetros nos vetores da lista.

```
1  # Configurações do treinamento - (seed, numberOfLines, dataDivisionMethod)
2  # Uma análise
3  analysis1<-c(123, "all", "createDP")
4
5  # Mais de uma análise
6  # analysis2<-c(1234, "all", "createDP")
7  # analysis3<-c(123, 2000, "createDP")
8
9  # Juntando os vetores das configurações escolhidas na lista
10 # Um vetor de parâmetro
11 analysisList<-list(analysis1)
12
13 # Mais de um vetor de parâmetros
14 # analysisList<-list(analysis1, analysis2, analysis3)
```

Código 3.5 – Configuração da lista de vetores com os parâmetros esclhidos para análise de treinamento

Com a escolha do *dataset* e do número de analise desejadas a execução do algoritmo pode ser iniciada. Na sua execução ocorre algumas configurações e ações para realizar a análise. Isto é, ao iniciar o algoritmo, primeiro é feito configurações iniciais, que estão relacionadas a criação de funções que o algoritmo vai precisar usar. Depois é lido o *dataset* especificado e em seguida a lista com os vetores dos parâmetros de análise. Com essa informações, no próximo passo o algoritmo verifica se o *dataset* especificado existe ou não, se não existir ele informa que o mesmo não existe e termina a sua execução. Entretanto, caso o *dataset* exista, ele executa a próxima etapa iniciando uma estrutura de repetição *For* que vai verificar o número de vetores da lista e a partir desse valor, vai realizar as ações necessárias para o treinamento dos modelos. Dessa forma, com a lista possuindo um vetor, será realizado uma análise de treinamento, caso ela possua dois ou mais será realizado a análise correspondentes a esses números. Caso não se tenha nenhum vetor, a lista é considerada de tamanho zero e a execução do algoritmo é finalizada.

Com a lista possuindo algum vetor, então a estrutura de repetição *For* é iniciada e os parâmetros informados no primeiro vetor são lidos. Com a leitura dos parâmetros, primeiro é utilizado o parâmetro *seed* pela função *set.seed(seed)* para que os resultados obtidos da análise possam ser replicados por outra pessoa. Após essa definição, é usado o segundo parâmetro e verificado qual o número de amsotras do *dataset* deve-se usar. Se o valor for o '*all*' então será usado todas as amostras do *dataset*, senão se usará o número de amostras informado. Com a escolha do número de amostra, no próximos passo é feita outras configurações necessárias, que envolvem criar outras funções usadas pelo algoritmo. Feito essa configurações, na próxima etapa é utilizado o terceiro e último parâmetro para verificar qual função deve-se usar para dividir os dados do *dataset*. Se for '*createDP*' então usa-se a função *createDataPartition*, senão usa-se a função *sample()*. Terminada essas definições com os parâmetros passados, na próxima etapa é realizado o treinamento com as técnicas selecionas nesse trabalho usando a função *train* com os dados de treino. Terminado o treinamento, no passo seguinte é feito a classificação dos melhores modelos em relação a acurácia. Depois dessa classificação, é realizado a predição dos melhores modelos definidos na etapa anterior com o uso da matriz de confusão descrita na subseção 2.6.3 através da função *confusionMatrix* usando os dados de teste. E por último os dados obtidos são salvos em arquivos *.csv* para fazer a análise dos resultados. Terminado todo esse processo, com o primeiro vetor de parâmetros informado, caso se tenha outros, é realizado a próxima analise até que se tenha lido todos os vetores passados. Quando todos forem lidos e processados a execução do algoritmo termina e os resultados podem ser verificados nos arquivos *.csv* criados.

A partir dessa estrutura de execução do algoritmo, nas subseções a seguir é apresentado de modo geral as configurações realizadas no código do algoritmo para o uso das funções *createDataPartition*, *train* e *confusionMatrix*.

3.4.3 Configuração da função *createDataPartition*

No algoritmo a implementação da função *createDataPartition* foi feita como mostrado no código 3.6. Além da função *createDataPartition*, foi adicionado também o uso da função *sample* nativa da linguagem R, para embaralhar as amostras nos conjuntos de dados de treino e teste gerados. No uso da função *createDataPartition*, os índices gerados já são feitos aleatoriamente, no entanto, o dataset usado estava composto com a primeira metade das amostras com vozes masculinas e a segunda metade com vozes femininas. Desse modo, mesmo com o uso da função *createDataPartition* os índices resultantes apresentavam sequencias de vozes masculinas e femininas nos conjuntos. Assim para ficar o menos sequencial possível, se utilizou a função *sample* para embaralhar esses conjuntos.

```
1  if (dataDivisionMethod == "createDP"){
2        # Gerando valores aleatórios de indices com 80% das amostras do dataset
3        trainingDataIndex  <- createDataPartition(dfGenderVoice$label, p=0.8, list=
     FALSE)
4
5        # Pega todas as linhas que está definida em "trainingDataIndex"
6        trainingData <- dfGenderVoice[ trainingDataIndex, ]
7
8        # Embaralhando as linhas das amostras de treino
9        trainingData <- trainingData[sample(nrow(trainingData)),]
10
11       # Pega todas as linhas que não está definida em "trainingDataIndex", o sinal
     de - (menos) diz pra não pegar essas linhas
12       testData  <- dfGenderVoice[-trainingDataIndex, ]
13
14       # Embaralhando as linhas das amostras de teste
15       testData <- testData[sample(nrow(testData)),]
16     } else {
17       grupos <- sample(2, nrow(dfGenderVoice), replace=TRUE, prob=c(0.8, 0.2))
18
19       ## Dividindo dados para treino e para teste
20       trainingData <- dfGenderVoice[grupos==1,]
21
22       # Embaralhando as linhas das amostras de treino
23       trainingData <- trainingData[sample(nrow(trainingData)),]
24
25       testData <- dfGenderVoice[grupos==2,]
26
27       # Embaralhando as linhas das amostras de teste
28       testData <- testData[sample(nrow(testData)),]
29     }
```

Código 3.6 – Configuração da função *createDataPartition* no algoritmo de treino desenvolvido

3.4.4 Configuração da função *train*

Na sua configuração no algoritmo se utilizou seis parâmetros dos totais fornecidos pela função, que foram considerados suficientes para realizar o treinamento das técnicas escolhidas, ficando assim a função declarada como mostrado no código 3.7.

```
1 modelTechnical <- train(form, data, method, trControl, metric, preProcess)
```

Código 3.7 – Parâmetros usados na função *train* para treinamento dos modelos

Para cada parâmetro foi necessário definir uma opção, ficando a função reescrita como mostrado no código 3.8:

```
1 modelTechnical <- train(y ~ ., data=chosen-data, method="chosen-method", trControl=
    fitControl, metric="chosen-metric", preProcess=c("chosen-preProcess"))
```

Código 3.8 – Opções definidas no parâmetros usados na função *train* para treinamento dos modelos

Além disso, no parâmetro *trControl* é preciso defini-lo pela função *trainControl*, o que foi feito através da variável *fitControl*. Na sua configuração também se usou seis argumentos dos totais disponibilizados por essa função, que foram considerados suficientes para a realização do treinamento, ficando a função declarada como mostrado no código 3.9.

```
1 fitControl <- trainControl(method = "chosen-method", number = chosen-number,
    repeats = chosen-number, savePredictions = chosen-savePredictions, classProbs =
     chosen-classProbs, returnResamp = "chosen-returnResamp")
```

Código 3.9 – Parâmetros da função *trainControl*

Abaixo são descritos a definição de cada parâmetro escolhido nas funções *train* e *trainControl*:

- **form** - Uma fórmula descrita por esse formato: *'y ∼ x1 + x2 + ...'* ou esse *'y ∼ .'* em que:

  x - Define as colunas a serem usadas para pegar as amostras do quadro de dados(dataframe) de treino.

  y - Define a coluna (um vetor numérico ou fatorial) contendo o resultado para cada amostra (coluna target) do quadro de dados(dataframe) de treino.

  . - O ponto representa todas as colunas do quadro de dados(dataframe) de treino menos a definida em y (coluna target).

- **data** - Quadro de dados(dataframe) do qual as variáveis especificadas em *form* são retiradas.

- **method** - Uma string que especifica qual modelo de classificação ou regressão usar. Atualmente 238 estão disponíveis.

- **trControl** - Uma lista de valores que definem o tipo de re-amostragem que a função *train* deve realizar. Essa lista é definida em *trainControl*.

- **trainControl** - Parâmetros que controlam o tipo de re-amostragem da função *train*. Varios parâmetros são disponibilizados, no entanto, só se usou seis deles que foram considerados necessários, sendo eles:

  - **method**: Determina o método de re-amostragem usado, tendo como opções: *boot*, *boot632*, *optimism_boot*, *boot_all*, *cv (cross validation)*, *repeatedcv (repeat cross validation)*, *LOOCV*, *LGOCV*(para divisões repetidas de treinamento/-teste), *none*(apenas ajusta um modelo para todo o conjunto de treinamento), *oob*(apenas para floresta aleatória, árvores ensacadas, terra ensacada, análise discriminante flexível ensacada ou modelos de floresta de árvores condicionais), *timeslice*, *adaptive_cv*, *adaptive_boot* e *adaptive_LGOCV*.

  - **number**: O número de grupos (folds) que o dataset será dividido representando o número de iterações de re-amostragem.

  - **repeats**: Apenas usado para o method de validação cruzada de k grupos (fold) repetidos (*repeatedcv*). O qual representa o número de conjuntos completos de folds para calcular.

  - **savePredictions**: Habilita que as previsões individuais feitas durante a validação cruzada sejam salvas e visualizadas. Os termos possíveis podem ser *"all"*, para salvar todas as previsões individuais, *"none"* para não salvar e *"final"* para salvar as previsões para os parâmetros de ajuste ideais. Um valor lógico (T - TRUE) também pode ser usado para definir o termo *"all"* e (F - FALSE) para o termo *"none"*.

  - **classProbs**: Um valor lógico (T - TRUE ou F - FALSE) que define se as probabilidades das classes da coluna target(label) do dataframe devem ser calculadas em cada re-amostragem, junto com os valores já previstos para medir o desempenho. Com esse parâmetro ativo (TRUE), isso vai mesclar colunas de probabilidades nas previsões geradas a partir de cada re-amostragem(colunas com os nomes das classes serão adicionadas contendo as probabilidades da classe). O seu uso deve ser feito quando utilizado modelos de classificação.

  - **returnResamp**: Habilita informações sobre as re-amostras. Pode-se escolher três opções, sendo elas, *"none"* que não mostra nenhuma informação final, *"final"* que mostra as informações da re-amostragem final e *"all"* que mostra as informações sobre todas as re-amostragens.

- **metric** - Uma string que especifica qual métrica de resumo será usada para selecionar o modelo ideal. Por padrão, os valores possíveis são *RMSE* e *Rsquared* para regressão e *Accuracy* e *Kappa* para classificação. Se métricas de desempenho personalizadas forem usadas através do argumento *summaryFunction* em *trainControl*, o valor de **metric** deve corresponder a um dos argumentos. Se não corresponder, um aviso será emitido e a primeira métrica fornecida por o *summaryFunction* será usada.

- **preProcess** - Um vetor de string que define um pré-processamento dos dados do preditor. Isto é, a função estima o que for necessário do conjunto de dados selecionado, no caso o conjunto de treinamento, e em seguida aplica essas transformações sem recalcular os valores As possibilidades atuais são *BoxCox*, *YeoJohnson*, *expoTrans*, *center*, *scale*, *range*, *knnImpute*, *bagImpute*, *medianImpute*, *pca*, *ica* e *spatialSign* . Caso não seja definido um modelo o padrão é sem pré-processamento. O código de pré-processamento só é projetado para funcionar quando x é uma matriz simples ou quadro de dados(dataframe).

Verificado as possíveis opções oferecidas pelos argumentos escolhidos para uso nas funções, a escolha das opções desejadas se fez considerando as que atendiam o cenário proposto. Desse modo, para o parâmetro *trControl* todas as seis técnicas escolhidas receberam a mesma configuração, como mostrado no código 3.10.

```
1 fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 3,
      savePredictions = T, classProbs = T, returnResamp = "all")
```

Código 3.10 – Configuração do fitControl para as seis técnicas escolhidas

Nessa configuração se definiu que o método de re-amostragem usado é o de validação cruzada com repetição, onde o número de grupos(folds) para as iterações de re-amostragem seja igual a 10 e que o processo seja repetido 3 vezes. Essa escolha foi feita com base no que foi apresentado nas subseções 2.6.1 e 2.6.2, em que para aproveitar um número maior de dados do conjunto de dados para o treino do modelo, o uso da técnica validação cruzada é uma das opções recomendadas, por permitir que através do uso do conjunto de treinamento já se treine e avalie o resultado dos modelos através do método validação cruzada *k-fold*, e com isso já se tenha um melhor resultado do treinamento e detecção se os mesmos possam apresentar sobreajuste. Além disso, também se habilitou que as previsões individuais feitas durante a validação cruzada sejam salvas e visualizadas, para auxiliar na verificação do resultados, que as probabilidades das classes sejam previstas no processo de re-amostragem, para ajudar na medição do desempenho e que as informações sobre todas as re-amostragens sejam mostradas, para novamente auxiliar na verificação dos resultados.

Definido o *fitControl*, para a escolha dos demais parâmetros da função *train*, os mesmos tiveram a maioria dos parâmetros iguais, só se tendo como mudança o parâmetro

method que se alterou para cada técnica de aprendizado escolhida, como mostrado no código 3.11.

```
1 modelTechnical <- train(label ~ ., data=trainingData, method="chosen-method",
    trControl=fitControl, metric="Accuracy", preProcess=c("center","scale"))
```

Código 3.11 – Definição dos parâmetros da função *train* para cada técnica

Na interpretação de sua configuração, no primeiro parâmetro se escolheu a coluna *label* do dataframe para definir o resultado para cada amostra do quadro de dados(dataframe) de treino, e as demais colunas para serem usadas como variáveis para a criação do modelo, representada pelo carácter "."(ponto). No segundo parâmetro se definiu o dataframe de treino (80% dos dados) onde será acessado as informações para a criação do modelo de treinamento. No terceiro parâmetro foi definido as técnicas escolhidas como mostrado na tabela 6. No quarto parâmetro se configurou o *trControl* pela variável *fitControl* já descrito acima para configurar a criação do modelo com re-amostragem. No quinto parâmetro se especificou a métrica para selecionar o modelo ideal como sendo *Accuracy* (Acurácia), por analisar a exatidão do resultado do modelo e também pelo método de treino escolhido ser de classificação. E no sexto e último parâmetro se definiu um pré-processamento nos dados de predição como sendo *center*, que subtrai a média dos dados pelos valores dos dados e *scale*, que divide os valores pelo desvio padrão, para respectivamente estimar a localização e escalas dos preditores, e com isso ajudar no resultado da aprendizagem.

Como mencionado antes, os parâmetros configurados para cada técnica na função *train* foram iguais para todas elas, só se alterando a técnica escolhida no parâmetro *method*. Na escolha das técnicas para fazer o treinamento, o *caret* oferece várias técnicas para se escolher, sendo que de cada técnica disponibilizada, a mesma possui mais de uma variação de sua implementação, apresentando como diferença a sua lógica ou o número de parâmetros de ajustes. Isto é, se utilizar a técnica de ANN, a mesma possui mais de 10 variações de sua implementação no caret, como por exemplo *Neural Network*, *Neural Network Multi-Layer Perceptron*, *Neural Network Multi-Layer Perceptron with multiple layers*, entre outras. Assim, para as outras técnicas disponibilizadas, ocorre a mesma variação. Desse modo, para a escolha das técnicas no parâmetro *method*, se selecionou os métodos base de cada técnica escolhida nesse trabalho e também mais duas variações da mesma, sendo que as que não possuíam duas, foi escolhido só uma, com o objetivo de verificar o quanto o resultado do aprendizado se altera em relação a isso e já servindo também como escolha para a análise do desempenho. Na tabela 6 é apresentado os métodos escolhidos do parâmetro *method* para a função *train* em cada técnica e a nomenclatura usada para identificar posteriormente os resultados dos modelos.

Na tabela 7 é mostrado novamente os métodos escolhidos, mas com alguns detalhes a mais, como o nome do modelo do método, o tipo de treinamento que ele pode ser usado,

Tabela 6 – Técnicas e métodos usados na função *train* para o treinamento

Algoritmo		**Método**	**Nomenclatura da técnica analisada**
Artificial Neural Networks - ANN		nnet	ANN 1 (nnet)
		mlp	ANN 2 (mlp)
		mlpML	ANN 3 (mlpML)
Support Vector Machines - SVM		svmRadial	SVM 1 (svmRadial)
		svmRadialWeights	SVM 2 (svmRadialWeights)
		svmPoly	SVM 3 (svmPoly)
Decision Trees - DT	CART	rpart	CART 1 (rpart)
		rpart1SE	CART 2 (rpart1SE)
		rpart2	CART 3 (rpart2)
	C5.0	C5.0	DecisionTrees 1 (C5.0)
		C5.0Rules	DecisionTrees 2 (C5.0Rules)
		C5.0Tree	DecisionTrees 3 (C5.0Tree)
Random Forest - RF		rf	Random Forest 1 (rf)
		cforest	Random Forest 2 (cforest)
		parRF	Random Forest 3 (parRF)
k-Nearest Neighbors - K-NN		knn	k-NN 1 (knn)
		kknn	k-NN 2 (kknn)
Naive Bayes - NB		nb	Naive Bayes 1 (nb)
		naive_bayes	Naive Bayes 2 (naivebayes)

FONTE: Elaborada pelo autor.

a biblioteca utilizada pelo *caret* para executar o método e os parâmetros de ajustes que o modelo oferece para se obter mudanças no resultado do desempenho no treinamento do modelo. Sobre essas informações, as que apresentam maior interesse são os *Parâmetros de ajustes*, pois através da alteração de seus valores pode-se ir verificando quais apresentam melhores resultados no desempenho, e com isso um método que estava melhor em relação a outro, pode-se ter uma redução ou ficar ainda melhor no seu resultado em relação ao outro que estava pior, tornando possível se conseguir vários resultados. Como alterar os *Parâmetros de ajuste* irá resultar em vários possíveis resultados e já se está analisando 19 métodos das técnicas escolhidas, caso se utilize os parâmetros para se verificar novos resultados de desempenho, isso resultará em muitas análises e não se terá um padrão na verificação das técnicas. Desse modo, nessa análise se decidiu não utilizar valores específicos nos parâmetros, usando dessa forma os valores padrões que já vem configurados no uso dos métodos. Isso representa em não colocar nenhuma informação nos parâmetros de ajuste, o que resulta em uma estrutura padronizada na análise. Por adotar essa escolha, sebe-se que nesse cenário vai se verificar que a técnica X apresente o melhor desempenho. Mas caso se altere os parâmetros de ajuste, essa técnica pode já não ser mais a melhor, se tendo dessa situação, uma proposta de análise para trabalhos futuros.

Tabela 7 – Algoritmos usados para o treinamento - mais detalhes

Modelo	**Método**	**Tipo**	**Biblioteca**	**Parâmetros de ajuste**
Neural Network	nnet	Classification, Regression	nnet	size, decay
Multi-Layer Perceptron	mlp	Classification, Regression	RSNNS	size
Multi-Layer Perceptron with multiple layers	mlpML	Classification, Regression	RSNNS	layer1, layer2, layer3
Support Vector Machines with Radial Basis Function Kernel	svmRadial	Classification, Regression	kernlab	sigma, C
Support Vector Machines with Class Weights	svmRadialWeights	Classification	kernlab	sigma, C, Weight
Support Vector Machines with Polynomial Kernel	svmPoly	Classification, Regression	kernlab	degree, scale, C
CART	rpart	Classification, Regression	rpart	cp
CART	rpart1SE	Classification, Regression	rpart	None
CART	rpart2	Classification, Regression	rpart	maxdepth
C5.0	C5.0	Classification	C50, plyr	trials, model, winnow
Single C5.0 Ruleset	C5.0Rules	Classification	C50	None
Single C5.0 Tree	C5.0Tree	Classification	C50	None
Random Forest	rf	Classification, Regression	randomForest	mtry
Conditional Inference Random Forest	cforest	Classification, Regression	party	mtry
Parallel Random Forest	parRF	Classification, Regression	e1071, randomForest, foreach, import	mtry
k-Nearest Neighbors	knn	Classification, Regression		k
k-Nearest Neighbors	kknn	Classification, Regression	kknn	kmax, distance, kernel
Naive Bayes	nb	Classification	klaR	fL, adjust, usekernel
Naive Bayes	naive_bayes	Classification	naivebayes	laplace, adjust usekernel

FONTE: Elaborada pelo autor.

3.4.5 Configuração da função *confusionMatrix*

Na configuração da função *confusionMatrix* em um trecho do código foi feito a predição dos modelos para cada técnica, usando a função *predict* nativa da linguagem R. Essa função faz a previsão de valores com base em um objeto de modelo linear, ao se passar como argumentos o modelo treinado e os dados de teste sem a coluna label, como mostrado no código 3.12.

```
predictionAnn <- predict(modelAnnBestTechnique, testDataWithoutLabelColumn)
predictionSvm <- predict(modelSvmBestTechnique, testDataWithoutLabelColumn)
predictionCart <- predict(modelCartBestTechnique, testDataWithoutLabelColumn)
predictionDt <- predict(modelDtBestTechnique, testDataWithoutLabelColumn)
predictionRf <- predict(modelRfBestTechnique, testDataWithoutLabelColumn)
predictionNb <- predict(modelNbBestTechnique, testDataWithoutLabelColumn)
```

Código 3.12 – Trecho do código do algoritmo com as predições dos modelos treinados

A partir desses resultados as predições são passadas para a função *confusionMatrix* junto com a coluna label dos dados de teste para para fazer a avaliação das métricas com essas informações, como mostrado no código 3.13.

```
confusionMatrixAnn <- confusionMatrix(predictionAnn, testDataLabelColumn)
confusionMatrixSvm <- confusionMatrix(predictionSvm, testDataLabelColumn)
confusionMatrixCart <- confusionMatrix(predictionCart, testDataLabelColumn)
confusionMatrixDt <- confusionMatrix(predictionDt, testDataLabelColumn)
confusionMatrixRf <- confusionMatrix(predictionRf, testDataLabelColumn)
confusionMatrixKnn <- confusionMatrix(predictionKnn, testDataLabelColumn)
confusionMatrixNb <- confusionMatrix(predictionNb, testDataLabelColumn)
```

Código 3.13 – Trecho de código do algoritmo com o uso da função *confusionMatrix* nos modelos treinados

4 TESTES E RESULTADOS EXPERIMENTAIS

Neste capítulo é apresentado os resultados gerados pela utilização dos dados de treino e de teste nos modelos treinados no algoritmo desenvolvido.

4.1 Aplicação dos dados de treino nos modelos treinados

Na execução do algoritmo, se configurou a lista de vetores com somente um vetor contendo nos parâmetros *seed*, *numberOfLines* e *dataDivisionMethod* respectivamente os valores 123, *all* e *createDP*, como mostrado no código 4.1.

```
# Configurações do treinamento - (seed, numberOfLines, dataDivisionMethod)
analysis1<-c(123, "all", "createDP")

# Juntando os vetores das configurações escolhidas na lista
analysisList<-list(analysis1)
```

Código 4.1 – Configuração da lista contendo um vetor de parâmetros para o treinamento dos modelos

Após a execução, os resultados dos treinos são mostrados na tabela 8. Além desses resultados, são mostrados também informações adicionais, para se ter uma referência a que configurações os resultados do treino estavam associados. Dentre essas informações estão o tempo que demorou para o treinamento acontecer, a semente usada no treinamento, o número de amostras totais usadas, o número da divisão das amostras totais de vozes femininas e masculinas, o método usado para fazer a divisão das amostras, o número de amostras usadas para o treinamento, junto com a divisão dos números de amostras usadas de vozes femininas e masculinas desse conjunto, o número de amostras usadas no teste do modelo, junto também com a divisão das amostras de vozes femininas e masculinas desse conjunto e o sistema operacional usado para realizar o treino, como é mostrado na tabela 9.

Na tabela 8, como a métrica escolhida foi a acurácia, nos seus resultados são mostrados os parâmetros acurácia miníma (*Min_Accuraccy*), 1° quartil da acurácia (*1st.Qu_Accuracy*), mediana da acurácia (*Median_Accuraccy*), acurácia média (*Mean_Accuracy*), 3º quartil da acurácia (*3rd.Qu_Accuracy*) e acurácia máxima (*Max_Accuracy*). Desses parâmetros, o de interesse é a acurácia média (*Mean_Accuracy*), que determina o valor em porcentagem das previsões certas que o modelos fez. Desse modo, ao se filtrar os resultados

Tabela 8 – Resultado do treinamento das técnicas escolhidas com o uso da função *train*

Technique	Min Accuracy	1st.Qu Accuracy	Median Accuracy	Mean Accuracy	3rd.Qu Accuracy	Max Accuracy	Runtime training
ANN 1 (nnet)	0.9606	0.9734	0.9783	0.9769	0.9803	0.9961	00:00:41
ANN 2 (mlp)	0.9486	0.9684	0.9723	0.9712	0.9764	0.9921	00:00:42
ANN 3 (mlpML)	0.9484	0.9655	0.9724	0.9725	0.9764	0.9961	00:00:43
SVM 1 (svmRadial)	0.9605	0.9763	0.9802	0.9797	0.9842	1	00:00:20
SVM 2 (svmRadialWeights)	0.9682	0.9763	0.9803	0.9807	0.9872	0.9921	00:00:54
SVM 3 (svmPoly)	0.9567	0.9724	0.9763	0.9771	0.9842	0.9921	00:03:47
CART 1 (rpart)	0.9447	0.9526	0.9605	0.9616	0.9685	0.9803	00:00:01
CART 2 (rpart1SE)	0.9328	0.9526	0.9606	0.9603	0.9685	0.9882	00:00:02
CART 3 (rpart2)	0.9249	0.9526	0.9626	0.9616	0.9685	0.9842	00:00:01
Decision Trees 1 (C5.0)	0.9567	0.9724	0.9802	0.9787	0.9871	0.9921	00:00:59
Decision Trees 2 (C5.0Rules)	0.9488	0.9684	0.9724	0.9717	0.9763	0.9921	00:00:05
Decision Trees 3 (C5.0Tree)	0.9331	0.9645	0.9704	0.9692	0.9793	0.9842	00:00:05
Random Forest 1 (rf)	0.9646	0.9724	0.9802	0.9792	0.9872	0.9921	00:02:00
Random Forest 2 (cforest)	0.9567	0.9685	0.9763	0.9752	0.9803	0.9882	00:21:41
Random Forest 3 (parRF)	0.9567	0.9724	0.9803	0.9792	0.9842	0.9921	00:01:49
k-NN 1 (knn)	0.9565	0.9684	0.9724	0.9724	0.9793	0.9882	00:00:03
k-NN 2 (kknn)	0.9527	0.9723	0.9764	0.9761	0.9803	0.9882	00:00:46
Naive Bayes 1 (nb)	0.8735	0.9065	0.9212	0.9213	0.9331	0.9605	00:00:24
Naive Bayes 2 (naive_bayes)	0.8898	0.9092	0.9212	0.9201	0.9319	0.9449	00:00:02

FONTE: Elaborada pelo autor.

Tabela 9 – Configuração usada no treinamento das técnicas escolhidas

Seed	Samples	Female samples	Male samples	Data division method	Training samples 80%	Training samples female	Training samples male	Test samples 20%	Test samples female	Test samples male	Operational System
123	3168	1584	1584	createDP	2536	1268	1268	632	316	316	unix

FONTE: Elaborada pelo autor.

em relação a melhor acurácia média de cada técnica, se tem a classificação do desempenho da melhor a pior técnica verificada, como é mostrado na tabela 10.

Tabela 10 – Resultado da classificação do desempenho das técnicas escolhidas dos dados de treino

Technique	Mean_Accuracy	Runtime training
SVM 2 (svmRadialWeights)	98,07%	00:00:54
Random Forest 1 (rf)	97,92%	00:02:00
Decision Trees 1 (C5.0)	97,87%	00:00:59
ANN 1 (nnet)	97,69%	00:00:41
k-NN 2 (kknn)	97,61%	00:00:46
CART 3 (rpart2)	96,16%	00:00:01
Naive Bayes 1 (nb)	92,13%	00:00:24

FONTE: Elaborada pelo autor.

Analisando o resultado da tabela 10, se verificou que a melhor técnica foi a *SVM 2 (svmRadialWeights)* com 98,07% de acurácia para identificar as vozes femininas e masculinas, e a pior técnica foi a *Naive Bayes 1 (nb)* com 92,13%. Entre essas duas

as demais tiveram a sua acurácia variando entre 96,16% a 97,92% o que foram ótimos valores, mas o ideal é ficar o mais perto possível dos 100%, que foi o caso da *SVM 2 (svmRadialWeights)*. Outro ponto a se notar é que o tempo de execução das técnicas foi variado, e a que demorou mais tempo para realizar o treinamento não representa a que tem o melhor resultado. Como pode ser visto na técnica *CART 3 (rpart2)* que foi mais rápida que a *Naive Bayes 1 (nb)* levando 1 segundo para concluir o seu treinamento e a *Random Forest 1 (rf)* que foi a que mais demorou na sua execução, mais ficou atrás da *SVM 2 (svmRadialWeights)* que levou 54 segundos, menos da metade do seu tempo e tendo o melhor resultado.

Com a verificação desses resultados, já se poderia determinar qual foi a melhor técnica. No entanto esses resultados não representam ainda o resultado final de qual técnica tem o melhor desempenho, pois os testes para ver o desempenho foram feitos com os dados de treino usando a validação cruzada k-fold. Para verificar o real desempenho, deve-se usar os dados de teste, uma vez que esses dados os modelos treinados nunca viram. Permitindo dessa forma, verificar se o resultado do modelo está bom e se não está apresentando sobreajuste (Overfitting), que acontece quando o resultado do treino foi bom, mas o de teste ficou ruim. Assim, para verificar o verdadeiro desempenho das técnicas foram usados os dados de teste por meio da técnica matriz de confusão.

4.2 Aplicação dos dados de teste nos modelos treinados

Com os modelos treinados se aplicou os dados de teste nos mesmos para fazer a predição dos resultados usando matrizes de confusão. Nos resultados foram exibidos os valores das predições, os valores totais das classes de referência e prevista e o número de dados totais dos dados de teste usados, como já mostrado na tabela 4. Com esses valores foram calculados suas métricas para avaliação do desempenho dos modelos, como mostrado nas subseções a seguir.

4.2.1 Modelo de Artificial Neural Networks - ANN 1 (nnet)

Na matriz de confusão apresentado na tabela 11 é mostrado as previsões feitas pelo modelo.

A partir desses resultados foram realizados os cálculos das métricas avaliativas do modelo, como demostrado nas equações 4.1 a 4.5. E na tabela 12 os resultados são organizados e exibidos.

$$Acurácia = \frac{307 + 308}{307 + 308 + 9 + 8} = \frac{615}{632} = 0,973101265 = 97,31\% \tag{4.1}$$

Tabela 11 – Matriz de confusão do modelo ANN 1 (nnet)

		Classes de referência		Total
		female voice	**male voice**	**previsto**
Classes previstas	**female voice**	307	8	315
	male voice	9	308	317
Total referência		316	316	632

FONTE: Elaborada pelo autor.

$$Taxa\ de\ erro = 1 - 0,973101265 = 0,026898734 = 2,69\% \tag{4.2}$$

$$Sensibilidade = \frac{307}{307+9} = \frac{307}{316} = 0,971518987 = 97,15\% \tag{4.3}$$

$$Especificidade = \frac{308}{308+8} = \frac{308}{316} = 0,974683544 = 97,47\% \tag{4.4}$$

$$Eficiência = \frac{97,15+97,47}{2} = \frac{194,62}{2} = 97,31\% \tag{4.5}$$

Tabela 12 – Resultado das métricas do modelo ANN 1 (nnet)

Métrica	**Resultado**
Acurácia	97,31%
Taxa de erro	2,69%
Sensibilidade	97,15%
Especificidade	97,47%
Eficiência	97,31%

FONTE: Elaborada pelo autor.

4.2.2 Support Vector Machines - SVM 2 (svmRadialWeights)

Na matriz de confusão apresentado na tabela 13 é mostrado as previsões feitas pelo modelo.

A partir desses resultados foram realizados os cálculos das métricas avaliativas do modelo, como mostrado na tabela 14.

4.2.3 Modelo Decision Trees 1 (C5.0)

Na matriz de confusão apresentado na tabela 15 é mostrado as previsões feitas pelo modelo.

Tabela 13 – Matriz de confusão do modelo SVM 2 (svmRadialWeights)

		Classes de referência		Total
		female voice	**male voice**	**previsto**
Classes previstas	**female voice**	309	4	313
	male voice	7	312	319
Total referência		316	316	632

FONTE: Elaborada pelo autor.

Tabela 14 – Resultado das métricas do modelo SVM 2 (svmRadialWeights)

Métrica	**Resultado**
Acurácia	98,26%
Taxa de erro	1,74%
Sensibilidade	97,78%
Especificidade	98,73%
Eficiência	98,26%

FONTE: Elaborada pelo autor.

Tabela 15 – Matriz de confusão do modelo Decision Trees 1 (C5.0)

		Classes de referência		Total
		female voice	**male voice**	**previsto**
Classes previstas	**female voice**	310	9	319
	male voice	6	307	313
Total referência		316	316	632

FONTE: Elaborada pelo autor.

A partir desses resultados foram realizados os cálculos das métricas avaliativas do modelo, como mostrado na tabela 16

Tabela 16 – Resultado das métricas do modelo Decision Trees 1 (C5.0)

Métrica	**Resultado**
Acurácia	98,1%
Taxa de erro	1,9%
Sensibilidade	98,1%
Especificidade	98,1%
Eficiência	98,1%

FONTE: Elaborada pelo autor.

4.2.4 Modelo Decision Trees - CART 3 (rpart2)

Na matriz de confusão apresentado na tabela 17 é mostrado as previsões feitas pelo modelo.

Tabela 17 – Matriz de confusão do modelo Decision Trees - CART 3 (rpart2)

		Classes de referência		**Total**
		female voice	**male voice**	**previsto**
Classes previstas	**female voice**	309	21	330
	male voice	7	295	302
Total referência		316	316	632

FONTE: Elaborada pelo autor.

A partir desses resultados foram realizados os cálculos das métricas avaliativas do modelo, como mostrado na tabela 18

Tabela 18 – Resultado das métricas do modelo CART 3 (rpart2)

Métrica	**Resultado**
Acurácia	95,57%
Taxa de erro	4,43%
Sensibilidade	97,78%
Especificidade	93,35%
Eficiência	95,57%

FONTE: Elaborada pelo autor.

4.2.5 Modelo Random Forest 1 (rf)

Na matriz de confusão apresentado na tabela 19 é mostrado as previsões feitas pelo modelo.

Tabela 19 – Matriz de confusão do modelo Random Forest 1 (rf)

		Classes de referência		**Total**
		female voice	**male voice**	**previsto**
Classes previstas	**female voice**	310	6	316
	male voice	6	310	316
Total referência		316	316	632

FONTE: Elaborada pelo autor.

A partir desses resultados foram realizados os cálculos das métricas avaliativas do modelo, como mostrado na tabela 20

Tabela 20 – Resultado das métricas do modelo Random Forest 1 (rf)

Métrica	Resultado
Acurácia	98,1%
Taxa de erro	1,9%
Sensibilidade	98,1%
Especificidade	98,1%
Eficiência	98,1%

FONTE: Elaborada pelo autor.

4.2.6 Modelo Naive Bayes 1 (nb)

Na matriz de confusão apresentado na tabela 21 é mostrado as previsões feitas pelo modelo.

Tabela 21 – Matriz de confusão do modelo Naive Bayes 1 (nb)

		Classes de referência		Total
		female voice	male voice	previsto
Classes previstas	female voice	278	12	290
	male voice	38	304	342
Total referência		316	316	632

FONTE: Elaborada pelo autor.

A partir desses resultados foram realizados os cálculos das métricas avaliativas do modelo, como mostrado na tabela 22

Tabela 22 – Resultado das métricas do modelo Naive Bayes 1 (nb)

Métrica	Resultado
Acurácia	92,09%
Taxa de erro	7,91%
Sensibilidade	87,97%
Especificidade	96,2%
Eficiência	92,09%

FONTE: Elaborada pelo autor.

4.2.7 Modelo k-NN 2 (kknn)

Na matriz de confusão apresentado na tabela 23 é mostrado as previsões feitas pelo modelo.

A partir desses resultados foram realizados os cálculos das métricas avaliativas do modelo, como mostrado na tabela 24

Tabela 23 – Matriz de confusão do modelo k-NN 2 (kknn)

		Classes de referência		Total previsto
		female voice	**male voice**	
Classes previstas	**female voice**	307	9	316
	male voice	9	307	316
Total referência		316	316	632

FONTE: Elaborada pelo autor.

Tabela 24 – Resultado das métricas do modelo k-NN 2 (kknn)

Métrica	Resultado
Acurácia	97,15%
Taxa de erro	2,85%
Sensibilidade	97,15%
Especificidade	97,15%
Eficiência	97,15%

FONTE: Elaborada pelo autor.

4.3 Análise dos Resultados

Com base nos resultados obtidos das métricas dos modelos na seção 4.2, nessa seção são apresentadas as análises comparativas das métricas em relação a acurácia, sensibilidade, especificidade e eficiência.

4.3.1 Comparação dos resultados dos modelos em relação a acurácia

Como já definido, a acurácia mede quantas previsões certas o modelo fez das classes verificadas em relação ao número total de amostras de testes aplicadas ao modelo para as previsões. Usando essa métrica o desempenho do modelo é verificado.

Desse modo, na tabela 25 é apresentado o resultado da acurácia dos modelos já ordenados do maior para o menor. Desses valores se verificou que a técnica que apresentou o melhor desempenho foi a "SVM 2 (svmRadialWeights)"com 98,26% de exatidão em suas previsões e que a técnica "Naive Bayes 1 (nb)"foi a técnica que teve o pior desempenho com 92,09%. Em relação as demais, a "CART 3 (rpart2)"obteve 95,57%, que foi um bom resultado. A "k-NN 2 (kknn)", "ANN 1 (nnet)"e "Decision Trees 1 (C5.0)"obtiveram respectivamente os valores 97,15%, 97,31% e 97,63%, que foram ótimos valores, podendo-se escolher qualquer uma dessas técnicas para uso, se considerando o valor de 97%, mas se for necessário escolher uma delas, o "Decision Trees 1 (C5.0)"seria o selecionado, pois no detalhe da sua porcentagem, foi o que teve melhor resultado. E por último o "Random Forest 1 (rf)"apresentou 98,1% que considerando só como 98% estaria no mesmo resultado que o

"SVM 2 (svmRadialWeights)", podendo-se escolher qualquer um deles, que o resultado esperado seria o mesmo. Mas novamente, caso se tenha que escolher qual usar , o "SVM 2 (svmRadialWeights)"seria o escolhido, por sua porcentagem ter sido um pouco melhor, e com isso ele foi considerado a técnica com melhor desempenho para a classificação das classes das vozes de gênero.

Tabela 25 – Resultado da métrica acurácia das técnicas com os dados de teste

Técnica	Acurácia
SVM 2 (svmRadialWeights)	98,26%
Random Forest 1 (rf)	98,1%
Decision Trees 1 (C5.0)	97,63%
ANN 1 (nnet)	97,31%
k-NN 2 (kknn)	97,15%
CART 3 (rpart2)	95,57%
Naive Bayes 1 (nb)	92,09%

FONTE: Elaborada pelo autor.

Comparando esses resultados com os resultados obtidos durante os treinos dos modelos, se verificou que a ordem de desempenho das técnicas continuou a mesma como mostrado na tabela 26. Além disso, se notou que a técnica com o melhor desempenho a "SVM 2 (svmRadialWeights)"teve uma pequena melhora. A com pior desempenho, a "Naive Bayes 1 (nb)"e as técnicas CART 3 (rpart2), k-NN 2 (kknn), ANN 1 (nnet) e Decision Trees 1 (C5.0) tiveram uma pequena redução no seu desempenho. E que a Random Forest 1 (rf), ao contrario dessas teve uma significativa melhora, mas não chegando a passar do resultado da SVM 2 (svmRadialWeights), que se consolidou como a melhor técnica a ser usar no reconhecimento das vozes femininas e masculinas.

Tabela 26 – Comparação da acurácia dos dados de teste com os dados de treino

Técnica	Acurácia dados de teste	Acurácia dados de treino
SVM 2 (svmRadialWeights)	98,26%	98,07%
Random Forest 1 (rf)	98,1%	97,92%
Decision Trees 1 (C5.0)	97,63%	97,87%
ANN 1 (nnet)	97,31%	97,69%
k-NN 2 (kknn)	97,15%	97,61%
CART 3 (rpart2)	95,57%	96,16%
Naive Bayes 1 (nb)	92,09%	92,13%

FONTE: Elaborada pelo autor.

4.3.2 Comparação dos resultados dos modelos em relação a sensibilidade, especificidade e eficiência

Na verificação das métricas do modelo, além de examinar a acurácia, pode-se também analisar a sensibilidade e especificidade para que possuindo esses resultados se possa averiguar a eficiência do modelo. Como já definido a sensibilidade exibe quantas previsões positivas o modelo fez corretamente, que no caso são as amostras com vozes femininas em relação ao número total de amostras de vozes femininas fornecidas ao modelo. A especificidade define o número de previsões certas de amostras não positivas, que no caso são as vozes masculinas em relação ao número total de amostras de vozes masculinas. E a eficiência define a média da soma dessas duas métricas. Com base nisso na tabela 27 é exibido o agrupamento dos resultados dessas métricas mostradas na seção 4.3.1 já seguindo a ordem de qual técnica foi a mais eficiente.

Tabela 27 – Comparação da eficiência dos modelos com os dados de teste

Técnica	Sensibilidade	Especificidade	Eficiência
SVM 2 (svmRadialWeights)	97,78%	98,73%	98,26%
Random Forest 1 (rf)	98,1%	98,1%	98,1%
Decision Trees 1 (C5.0)	98,1%	97,15%	97,63%
ANN 1 (nnet)	97,15%	97,47%	97,31%
k-NN 2 (kknn)	97,15%	97,15%	97,15
CART 3 (rpart2)	97,78%	93,35%	95,57%
Naive Bayes 1 (nb)	87,97%	96,2%	92,09%

FONTE: Elaborada pelo autor.

Analisando os resultados se verificou que como na comparação da acurácia, a técnica *SVM 2 (svmRadialWeights)* foi novamente a melhor técnica, apresentando uma eficiência de 96,26% em relação as demais. O *Naive Bayes 1 (nb)* foi o menos eficiente com 92,09%. E as demais técnicas seguiram a mesma ordem de eficiência como na comparação da métrica acurácia. Caso se considere o valor da eficiência como 98%, então a técnica *Random Forest 1 (rf)* ficaria com o mesmo valor do *SVM 2 (svmRadialWeights)* sendo também escolhido como a técnica mais eficiente.

Uma outra análise feita em relação aos resultados está na situação em que se precise ter um maior acerto na previsão das vozes femininas ou masculinas. Nessa situação, caso seja necessário ter maior acerto na previsão da voz feminina que representa a classe positiva e é indicado pelo resultado da sensibilidade do modelo. Isso foi observado na tabela 28 onde é mostrado os resultados já ordenados em relação a sensibilidade e se observou que as técnicas mas sensíveis foram a *Random Forest 1 (rf)* e *Decision Trees 1 (C5.0)* com 98,1% de acertos, e a menos sensível foi a *Naive Bayes 1 (nb)* com 87,97%. As demais técnicas resultaram nos valores de 97,78%, 97,78%, 97,15% e 97,15% respectivamente para *SVM 2 (svmRadialWeights)*, *CART 3 (rpart2)*, *ANN 1 (nnet)* e *k-NN 2 (kknn)*, notando-se

como mudança a técnica *SVM 2 (svmRadialWeights)* que dessa vez não foi a que teve o melhor resultado.

Tabela 28 – Comparação da sensibilidade dos modelos com os dados de teste

Técnica	Sensibilidade	Especificidade	Eficiência
Random Forest 1 (rf)	98,1%	98,1%	98,1%
Decision Trees 1 (C5.0)	98,1%	97,15%	97,63%
SVM 2 (svmRadialWeights)	97,78%	98,73%	98,26%
CART 3 (rpart2)	97,78%	93,35%	95,57%
ANN 1 (nnet)	97,15%	97,47%	97,31%
k-NN 2 (kknn)	97,15%	97,15%	97,15
Naive Bayes 1 (nb)	87,97%	96,2%	92,09%

FONTE: Elaborada pelo autor.

Já no caso que se precise de um maior acerto na previsão de vozes masculinas, representado pela classe não positiva e indicado pelo resultado da especificidade. Isso foi feito na verificação da tabela 29 que exibe os resultados já ordenado a partir da especificidade e os mesmo mostraram que a melhor técnica foi novamente a *SVM 2 (svmRadialWeights)* com 98,73% de acerto e a pior foi a *CART 3 (rpart2)* com 93,35%. As demais apresentaram valores entre 98,1% a 96,2%, e se notou como mudança a técnica *Naive Bayes 1 (nb)* que não foi a que teve o resultado mais baixo.

Tabela 29 – Comparação da especificidade dos modelos com os dados de teste

Técnica	Sensibilidade	Especificidade	Eficiência
SVM 2 (svmRadialWeights)	97,78%	98,73%	98,26%
Random Forest 1 (rf)	98,1%	98,1%	98,1%
ANN 1 (nnet)	97,15%	97,47%	97,31%
Decision Trees 1 (C5.0)	98,1%	97,15%	97,63%
k-NN 2 (kknn)	97,15%	97,15%	97,15
Naive Bayes 1 (nb)	87,97%	96,2%	92,09%
CART 3 (rpart2)	97,78%	93,35%	95,57%

FONTE: Elaborada pelo autor.

4.4 Comparação dos resultados de acurácia dos modelos em relação ao número de dados usados para o treinamento

Verificado qual foi a melhor e pior técnica após o treinamento dos modelos utilizando um vetor na lista com os valores *123*, *all* e *creatDP* que respectivamente representa o valor da semente sendo 123, usar todas as amostras do dataset e a divisão ser feita com a função *createDataPartition*. Nessa seção é apresentado uma análise dos resultados do treinamento ao se utilizar números menores de amostas fornecidas pelo dataset, com o objetivo de

observar como o número de amostras usadas no treinamento afeta nos resultados de desempenho das técnicas e no acerto das previsões. Na análise feita, se treinou novamente os modelos com as 3168 amostras totais, mas também se usando 2000, 1000 e 500 amostras. A semente (seed) e o método para a divisão dos dados continuaram sendo respectivamente 123 e *createDP*, porém na verificação com as 3168 amostras totais se fez uma nova análise, mas se alterando a semente para 1234, para ver a diferença que isso ocasiona. No código 4.2 é mostrado a configuração definida na lista de vetores do algoritmo para essa avaliação e na tabela 30 os resultados obtidos.

```
# Configurações do treinamento - (seed, numberOfLines, dataDivisionMethod)
analysis1<-c(123, "all", "createDP")
analysis2<-c(1234, "all", "createDP")
analysis3-c(123, 2000, "createDP")
analysis4-c(123, 1000, "createDP")
analysis5-c(123, 500, "createDP")

# Juntando os vetores das configurações escolhidas na lista
analysisList<-list(analysis1, analysis2, analysis3, analysis4, analysis5)
```

Código 4.2 – Configurações do treinamento para análise de número de amostras

Através desses resultados se observou que:

- Em todas as análises feitas a técnica Naive Bayes foi a que teve o pior desempenho, sendo que na verificação com 1000 amostras foi a Naive Bayes 2 (naive bayes) e nas demais foi a Naive Bayes 1 (nb).

- Com o uso da semente sendo 123, na análise usando 500 amostras a melhor técnica foi a *Decision Trees 1 (C5.0)* com 100% de exatidão enquanto as demais ficaram entre os valores de 99% a 90%. Na análise usando 1000 amostras a melhor técnica foi a *Random Forest 1 (rf)* com 97,5% de exatidão enquanto as demais ficaram entre os valores de 97% a 92,5%. Na análise usando 2000 amostras a melhor técnica foi a *Support Vector Machine 2 (svmRadialWeights)* com 98,75% de exatidão enquanto as demais ficaram entre os valores de 98% a 89,25%. E na análise usando 3168 amostras a melhor técnica foi a *Support Vector Machine 2 (svmRadialWeights)* com 98,26% de exatidão enquanto as demais ficaram entre os valores de 98,10% a 92,09%. Desses resultados, pode-se notar que usando um valor de semente fixo, mas com valores de amostras de dados diferentes, considerando o uso de um valor inicial e depois ir aumentando ele, o resultado da melhor técnica pode alterar, mas que a partir de uma certa quantidade de amostras utilizadas, o resultado da melhor técnica pode começar a ser o mesmo ou estar entre as duas melhores técnicas que tiveram os melhores desempenhos.

Tabela 30 – Comparação dos resultados de acurácia dos modelos em relação ao número de dados usados para o treinamento

Technique	Test data accuracy	Training data accuracy	Runtime Training	Seed	Samples	Training samples 80%	Test samples 20%
SVM 2 (svmRadialWeights)	98,26%	98,07%	00:00:54	123	3168	2536	632
RF 1 (rf)	98,10%	97,92%	00:02:00	123	3168	2536	632
DT 1 (C5.0)	97,63%	97,87%	00:00:59	123	3168	2536	632
ANN 1 (nnet)	97,31%	97,69%	00:00:41	123	3168	2536	632
k-NN 2 (kknn)	97,15%	97,61%	00:00:46	123	3168	2536	632
CART 3 (rpart2)	95,57%	96,16%	00:00:01	123	3168	2536	632
NB 1 (nb)	92,09%	92,13%	00:00:24	123	3168	2536	632
RF 3 (parRF)	97,94%	98,09%	00:01:52	1234	3168	2536	632
SVM 2 (svmRadialWeights)	97,63%	98,29%	00:00:55	1234	3168	2536	632
DT 1 (C5.0)	97,63%	97,95%	00:00:57	1234	3168	2536	632
ANN 1 (nnet)	97,15%	97,70%	00:00:41	1234	3168	2536	632
k-NN 2 (kknn)	96,84%	97,58%	00:00:47	1234	3168	2536	632
CART 1 (rpart)	96,20%	96,08%	00:00:01	1234	3168	2536	632
NB 1 (nb)	89,72%	92,01%	00:00:23	1234	3168	2536	632
SVM 2 (svmRadialWeights)	98,75%	97,21%	00:00:31	123	2000	1600	400
k-NN 2 (kknn)	98,00%	96,90%	00:00:29	123	2000	1600	400
DT 1 (C5.0)	97,75%	97,38%	00:00:38	123	2000	1600	400
RF 1 (rf)	97,75%	97,40%	00:01:01	123	2000	1600	400
ANN 1 (nnet)	97,50%	96,79%	00:00:24	123	2000	1600	400
CART 1 (rpart)	96,00%	95,65%	00:00:01	123	2000	1600	400
NB 1 (nb)	89,25%	91,69%	00:00:16	123	2000	1600	400
RF 1 (rf)	97,50%	97,00%	00:00:28	123	1000	800	200
CART 3 (rpart2)	97,00%	94,54%	00:00:01	123	1000	800	200
DT 1 (C5.0)	97,00%	97,13%	00:00:17	123	1000	800	200
SVM 3 (svmPoly)	96,50%	97,00%	00:00:46	123	1000	800	200
ANN 1 (nnet)	96,00%	96,96%	00:00:16	123	1000	800	200
k-NN 2 (kknn)	96,00%	95,58%	00:00:12	123	1000	800	200
NB 2 (naive_bayes)	92,50%	90,50%	00:00:02	123	1000	800	200
DT 1 (C5.0)	100,00%	95,92%	00:00:09	123	500	400	100
RF 3 (parRF)	99,00%	96,58%	00:00:13	123	500	400	100
SVM 1 (svmRadial)	96,00%	96,42%	00:00:03	123	500	400	100
k-NN 2 (kknn)	95,00%	96,00%	00:00:07	123	500	400	100
CART 3 (rpart2)	94,00%	93,67%	00:00:01	123	500	400	100
ANN 2 (mlp)	92,00%	96,25%	00:00:09	123	500	400	100
NB 1 (nb)	90,00%	89,08%	00:00:05	123	500	400	100

FONTE: Elaborada pelo autor.

- Na análise usando 3168 amostras e com a semente sendo 1234 a melhor técnica foi a *Random Forest 3 (parRF)* com 97,94% de exatidão enquanto as demais ficaram entre os valores de 97,63% a 89,72%. Sobre esse resultado, se notou que com a alteração da semente (seed), o número de amostras selecionadas para o treinamento também se alterou o que resultou na possível mudança de melhor técnica. No entanto como a segunda melhor técnica foi a *SVM 2 (svmRadialWeights)* com 97,63%, ao se considerar o valor de 97% ou arredondar os valores para 98%, a *SVM 2 (svmRadialWeights)* poderia ser novamente escolhida como melhor técnica. Além disso, desses resultados em relação a análise feita com a semente sendo o valor 123, a ordem dos desempenhos das técnicas foram quase as mesmas, só se alterando a técnica *CART* que nesse foi o 1 e o *Random Forest* que foi o 3 e sendo a melhor técnica.

- Em relação aos números de amostras usadas, no uso de 500 amostras a técnica com

melhor desempenho conseguiu 100% de acertos das previsões e as segunda 99%, sendo ótimos resultados. Porém, com o aumento das amostras, indo de 1000 para 2000 até os 3168 totais do dataset, se notou que o desempenho da acurácia teve uma leve redução, devido provavelmente a se ter mais amostras de exemplo no treinamento, o que adicionou possíveis novos valores dos parâmetros das amostras que não se tinha na verificação com 500 amostras. Desse modo, com mais possibilidades para se interpretar, dependendo como o modelo de treino foi configurado, a chances de previsões erradas ocorre é maior, resultando assim nessa diminuição do desempenho. Como no treinamento do modelo o ideal é se ter muitas amostras para ele aprender várias possibilidades na hora de realizar as previsões, e o resultado obtido da acurácia com o aumento dos números de amostras teve uma pequena diminuição nas previsões, se concluiu que as técnicas tiveram bons resultados no treinamento.

- Em relação ao tempo de treinamento dos modelos, se observou que com o aumento do números de amostras usados, o tempo de treino aumenta também, no entanto a técnica que demorou mais para realizar o treino não é necessariamente a de melhor desempenho, pois em algumas análises a melhor técnica teve o tempo de treino menor do que a segunda técnica, como pode ser visto nas análises de 500 e 3168 amostras com o valor da semente sendo 123.

Assim, de um modo geral em relação a esses resultados, se conclui que usar um valor considerado de amostras de dados é o mais recomendado, pois com poucas amostras o modelo pode apresentar uma acurácia de 100% ou próximo disso, mas se aparecer uma nova informação para previsão que o modelo não tenha visto, isso pode afetar o seu desempenho.

5 CONCLUSÕES

Neste trabalho foi apresentado um estudo para verificar o desempenho de técnicas de aprendizado de máquina pré-selecionadas, sendo elas *ANN*, *SVM*, *RF*, *DT*, *NB* e *K-NN*, para o reconhecimento do gênero de vozes, com o objetivo de determinar qual delas tem a melhor performance, para servir como um estudo de referência para uso em aplicações que necessitem utilizar esse recurso e também para comparações com técnicas recentes de *Deep Learning*.

Para o desenvolvimento do estudo foi preciso usar áudios de vozes femininas e masculinas para extrair os parâmetros que representam cada gênero de voz para serem utilizados nas técnicas de aprendizado de máquina, e assim realizar o treinamento. No entanto, como o foco do estudo estava voltado para os algoritmos de treinamento, para se ter esses parâmetros, se usou uma base de dados de gêneros de vozes já pronta com os parâmetros já extraídos, adquirida no site *data.world*.

Na criação do algoritmo de treinamento, para comparar o desempenho das técnicas escolhidas se utilizou a linguagem *R*, por possuir bibliotecas uteis, destinada para uso no contexto do aprendizado de máquina, incluindo tratamento dos dados e treinamento e validação dos modelos criados. E para facilitar o desenvolvimento do código se usou a IDE *RStudio* por inclui editor de código-fonte, automação de compilação local, debugger e outros recursos.

Com a definição dessas escolhas, os modelos de cada técnica escolhidas nesse estudo foram criados e treinados com o conjunto de treino, utilizando como técnica para avaliar os resultados a validação cruzada. Através do seu uso, os resultados dos desempenhos foram obtidos e uma filtragem da ordem da melhor a pior técnica foi realizada para determinar a melhor técnica. No entanto, esses resultados ainda não representavam a avaliação final dos resultados, pois foi utilizado os dados de treino nesse processo. Com isso, para validar realmente o resultado, foi utilizado os dados de teste nos modelos treinados e as predições feitas foram inseridas na técnica matriz de confusão que fornece as métricas avaliativas de desempenho dos modelos sendo elas a acurácia, sensibilidade, especificidade e eficiência.

Utilizando a matriz de confusão junto com as predições dos dados de teste, verificou-se quais foram as ordens de desempenho das técnicas analisadas em relação a acurácia e eficiência como foi mostrado respectivamente nas subseções 4.3.1 e 4.3.2. Dos resultados obtidos constatou-se que em relação a acurácia, principal métrica de desempenho, a técnica de melhor resultado foi o *SVM 2 (svmRadialWeights)* com 98,26% de exatidão e a pior foi o *Naive Bayes 1 (nb)* com 92,09%. E em relação a eficiência, a melhor técnica foi novamente o *SVM 2 (svmRadialWeights)* com 98,26% e a pior sendo outra vez o *Naive Bayes 1 (nb)*

com 92,09%. A partir desses resultados notou-se que por coincidência os resultados foram os mesmos para ambas as métricas.

Em relação ao número de amostras do dataset usado para o treinamento e teste dos modelos, se avaliou que no contexto desse trabalho ao se usar uma quantidade de amostras especificas, sendo neste cenário 3168 amostras, a melhor técnica verificada foi a *SVM2 (svmRadialWeights)*. No entanto, ao se avaliar o desempenho da melhor técnica com amostras menores que essa, se notou uma mudança nos resultados. Isto é, se avaliou os modelos com 2000, 1000 e 500 amostras com o valor de semente (*seed*) sendo 123 e com 3168 amostras mas com uma semente com o valor de 1234. Dos resultados obtidos, as com 500 e 1000 amostras tiveram respectivamente como melhores técnicas o *DT1 (C5.0)* e o *RF1 (rf)*. No resultado com 2000 amostras se teve a *SVM2 (svmRadialWeights)* novamente e com as 3168 amostras, mas com a semente sendo 1234, se teve a *RF3 (parRF)* como melhor técnica. Com base nesses resultados se considerou que a partir de um certo número de amostras, a técnica considerada a melhor fica mais evidente, o que pode ser visto com o desempenho das amostras de 2000 e 3168 com os valores de semente 123 e 1234. No caso das amostras com a semente com o valor 1234, a *SVM2 (svmRadialWeights)*, ficou em segundo lugar, mais ficando próximo do valor da melhor técnica. Já ao se utilizar amostras abaixo de 2000, as melhores técnicas se alteraram. O que dessa forma, pode influenciar na escolha da melhor técnica a ser usar.

Sobre os resultados obtidos da melhor a pior técnica, com base nas métricas, um ponto a se ressaltar é que esses resultados foram conseguidos a partir do uso da configuração padrão determinada nos parâmetros de ajuste de cada modelo de treino, do uso de todos os atributos do dataset e usando um número específico de amostras. No entanto, se essas configurações forem alteradas, os resultados obtidos podem permanecer os mesmos ou sofrer alterações, afetando o resultado da análise.

Desse modo, através do estudo proposto e dos resultados analisados, se concluiu que a técnica com o melhor desempenho foi a *SVM 2 (svmRadialWeights)*, sendo a indicada a ser usada em aplicações que precisem implementar o reconhecimento de vozes por gênero. A partir desse resultado e do processo de desenvolvimento desse trabalho, se pensou em algumas sugestões de projetos futuros propostos na seção a seguir.

5.1 Trabalhos Futuros

- Comparar os resultados desse trabalho com técnicas mais avançadas e atuais como os algoritmos de Deep Learning.

- Com base no resultado deste trabalho, alterar os parâmetros de ajustes de treino das técnicas utilizadas, e verificar qual o comportamento dos resultados;

- Criar um banco de áudios público de diferentes tipos de vozes femininas e masculinas;
- Através de um banco de áudios publico de vozes femininas e masculinas, executar a extração dos atributos usados na base de dados deste trabalho, e criar um novo conjunto de dados na extensão .csv para a aplicação e avaliação dos modelos propostos;
- Utilizar a técnica de aprendizado de máquina *Deep Learning* para avaliar o desempenho com base nesse estudo;
- Criar uma aplicação dividida na parte frontend e backend, onde no frontend deve conter uma tela para gravar a voz do usuário, e desse arquivo de áudio gerado, o mesmo deve ser enviado para uma API REST que representa o backend. No backend o áudio vai ser extraído nos atributos usados na base de dados deste trabalho para ser processado através do melhor modelo verificado também nesse trabalho. O resultado então vai ser devolvido para a parte frontend para informar ao usuário se a voz era feminina ou masculina.

REFERÊNCIAS

Ahmad, I. et al. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access*, v. 6, p. 33789–33795, 2018. ISSN 2169-3536. Citado 2 vezes nas páginas 42 e 43.

AMAZON. *Amazon Machine Learning*. 2016. Disponível em: <https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/machinelearning-dg.pdf>. Acesso em: 06 Julho. 2022. Citado 2 vezes nas páginas 50 e 51.

ARRIGONI, T. R. Reconhecimento de silhueta de automóveis para carros autônomos utilizando aprendizado de máquina. 2018. Citado 2 vezes nas páginas 38 e 46.

CASTANHEIRA, L. G. Aplicação de técnicas de mineração de dados em problemas de classificação de padrões. *Belo Horizonte: UFMG*, 2008. Citado na página 44.

CIPRIANO, J. L. G. Desenvolvimento de arquitetura para sistemas de reconhecimento automático de voz baseados em modelos ocultos de markov. 2001. Citado 3 vezes nas páginas 31, 32 e 33.

CIPRIANO, J. L. G. Técnicas para a implementação de sistemas de reconhecimento automático de voz. *Revista Tecnologia e Tendências - Universidade Feevale*, 2002. Citado 2 vezes nas páginas 24 e 31.

CN, P. *Confusion Matrix in R | A Complete Guide*. 2022. Disponível em: <https://www.digitalocean.com/community/tutorials/confusion-matrix-in-r>. Citado na página 53.

CONTI, J. P. J. et al. *Redes neurais recorrentes e expoente de Lyapunov aplicados a séries temporais financeiras*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2019. Citado na página 47.

DATA SCIENCE ACADEMY. *Capítulo 1 – Deep Learning e a Tempestade Perfeita*. 2019. Disponível em: <http://deeplearningbook.com.br/deep-learning-a-tempestade-perfeita>. Acesso em: 17 Outubro. 2019. Citado na página 23.

DATA SCIENCE ACADEMY. *Capítulo 4 – O Neurônio, Biológico e Matemático*. 2019. Disponível em: <http://deeplearningbook.com.br/o-neuronio-biologico-e-matematico>. Acesso em: 23 novembro. 2019. Citado 2 vezes nas páginas 39 e 41.

ENGLESON, S. *The Future of Voice From Smartphones to Smart Speakers to Smart Homes*. 2017. Disponível em: <https://www.comscore.com/Insights/Presentations-and-Whitepapers/2017/The-Future-of-Voice-From-Smartphones-to-Smart-Speakers-to-Smart-Homes>. Citado na página 24.

Fan, L. et al. Power-normalized plp (pnplp) feature for robust speech recognition. In: *2012 8th International Symposium on Chinese Spoken Language Processing*. [S.l.: s.n.], 2012. p. 224–228. Citado na página 24.

FERNANDES, T. G.; PANAZIO, A. N. Do analógico ao digital: amostragem, quantização e codificação. *II Simpósio de Iniciação Científica da Universidade Federal do ABC-SIC-UFABC*, 2009. Citado na página 32.

GABANINI, A. P. N. *A Voz Humana*. 2003. Disponível em: <http://www.profala.com/arttf57.htm>. Acesso em: 03 Novembro. 2019. Citado na página 29.

GEVERT, V. G. et al. Modelos de regressão logística, redes neurais e support vector machine (svm s) na análise de crédito a pessoas jurídicas. *RECEN-Revista Ciências Exatas e Naturais*, v. 12, n. 2, p. 269–293, 2010. Citado 2 vezes nas páginas 42 e 43.

GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1. Citado na página 25.

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, Amsterdam, v. 160, n. 1, p. 3–24, 2007. Citado na página 24.

KUHN, M. *The caret Package*. 2019. Disponível em: <https://topepo.github.io/caret>. Citado na página 59.

KUHN, M. *Model Training and Tuning*. 2019. Disponível em: <https://topepo.github.io/caret/model-training-and-tuning.html>. Citado na página 61.

Liu, S. et al. Random forest-based track initiation method. *The Journal of Engineering*, v. 2019, n. 19, p. 6175–6179, 2019. ISSN 2051-3305. Citado na página 46.

LORENA, A. C.; CARVALHO, A. C. de. Introduçaoas máquinas de vetores suporte. *Relatório Técnico do Instituto de Ciências Matemáticas e de Computaçao (USP/Sao Carlos)*, v. 192, 2003. Citado 2 vezes nas páginas 36 e 38.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. Citado na página 43.

MACHADO, M. L. Implementação de um sistema de reconhecimento automático de voz utilizando as técnicas mfcc e quantização vetorial com atributos dinâmicos, de normalização e detecção de voz ativa. Universidade Federal de Uberlândia, 2016. Citado 3 vezes nas páginas 25, 30 e 34.

MARIANO, D. *Métricas de avaliação em machine learning*. 2021. Disponível em: <https://diegomariano.com/metricas-de-avaliacao-em-machine-learning/>. Citado na página 54.

MARTINS, R.; YNOGUTI, C. Normalização do locutor em sistemas de reconhecimento de fala para usuários crianças. In: . [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 35 e 36.

MARTINS, R. M.; YNOGUTI, C. A. Normalizac ao do locutor em sistemas de reconhecimento de fala para usuarios criancas. 2014. Citado na página 35.

MCLOUGHLIN, I. *Applied speech and audio processing: with Matlab examples*. [S.l.]: Cambridge University Press, 2009. Citado 2 vezes nas páginas 29 e 30.

MEDIUM. *Aprendendo em uma Floresta Aleatória*. 2018. Disponível em: <https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleatória-3545f6babdf8>. Acesso em: 02 Dezembro. 2019. Citado 2 vezes nas páginas 46 e 47.

MEDIUM. *KNN (K-Nearest Neighbors) 1*. 2018. Disponível em: <https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e>. Acesso em: 01 Dezembro. 2019. Citado 3 vezes nas páginas 49, 50 e 52.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, v. 1, n. 1, p. 32, 2003. Citado na página 38.

MOTA, A. A. L. *Previsão de prêmio e a ocorrência de sinistros no mercado de seguro agrícola brasileiro*. Tese (Doutorado) — Universidade de São Paulo, 2019. Citado na página 46.

MUDA, L.; BEGAM, M.; ELAMVAZUTHI, I. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010. Citado na página 24.

NASCIMENTO, R. F. F. et al. O algoritmo support vector machines (svm): avaliação da separação ótima de classes em imagens ccd-cbers-2. *Simpósio Brasileiro de Sensoriamento Remoto*, v. 14, p. 2079–2086, 2009. Citado na página 42.

NUNES, H. F. et al. Reconhecimento de fala baseado em hmm. [sn], 1996. Citado 2 vezes nas páginas 33 e 34.

PATIL, T. R.; SHEREKAR, S. et al. Performance analysis of naive bayes and j48 classification algorithm for data classification. *International journal of computer science and applications*, v. 6, n. 2, p. 256–261, 2013. Citado na página 48.

PETRY, A.; ZANUZ, A.; BARONE, D. Reconhecimento automático de pessoas pela voz através de técnicas de processamento digital de sinais. *Instituto de Informática-Universidade Federal do Rio Grande do Sul*, 2000. Citado 3 vezes nas páginas 33, 34 e 36.

RAUBER, T. W. Redes neurais artificiais. *Universidade Federal do Espírito Santo*, 2005. Citado 3 vezes nas páginas 40, 41 e 42.

RDOCUMENTATION. *confusionMatrix: Create a confusion matrix*. 2022. Disponível em: <https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/confusionMatrix>. Citado na página 61.

RDOCUMENTATION. *createDataPartition: Data Splitting functions*. 2022. Disponível em: <https://www.rdocumentation.org/packages/caret/versions/6.0-92/topics/createDataPartition>. Citado na página 60.

RDOCUMENTATION. *train: Fit Predictive Models over Different Tuning Parameters*. 2022. Disponível em: <https://www.rdocumentation.org/packages/caret/versions/6.0-86/topics/train>. Citado na página 61.

Rivera-Lopez, R.; Canul-Reich, J. Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach. *IEEE Access*, v. 6, p. 5548–5563, 2018. ISSN 2169-3536. Citado na página 46.

RSTUDIO. *Download the RStudio IDE*. 2021. Disponível em: <https://www.rstudio.com/products/rstudio/download/>. Citado na página 58.

Shrawankar, U.; Thakare, V. Feature extraction for a speech recognition system in noisy environment: A study. In: *2010 Second International Conference on Computer Engineering and Applications*. [S.l.: s.n.], 2010. v. 1, p. 358–361. Citado na página 24.

SILVA, A. Reconhecimento de voz para palavras isoladas. *Universidade Federal de Pernanbuco*, 2009. Citado na página 34.

SILVA, L. M. Uma aplicação de árvores de decisão, redes neurais e knn para a identificação de modelos arma não-sazonais e sazonais. *Rio de Janeiro. 145p. Tese de Doutorado-Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro*, 2005. Citado 3 vezes nas páginas 45, 48 e 49.

TAKAKURA, A. M. et al. Uso do aprendizado de mÁquina no diagnÓstico mÉdico de patologias. *Colloquium Exactarum (Online)*, Universidade do Oeste Paulista, v. 10, n. 1, p. 78–89, 2018. ISSN COLLOQUIUM EXACTARUM. Disponível em: <https://doaj.org/article/147a15ae2357442681a83e46fceefef8>. Citado na página 23.

TORRES, J. Utilização do corte percentual na categorização de documentos da web com o algoritmo naive bayes. In: . [S.l.: s.n.], 2005. Citado na página 47.

VALIATI, J. F. Reconhecimento de voz para comandos de direcionamento por meio de redes neurais. 2000. Citado na página 29.

VERONEZ, M. et al. Estimativa de alturas geoidais para o estado de são paulo baseada em redes neurais artificiais. *Revista Brasileira de Geofísica*, v. 27, p. 583–593, 12 2009. Citado na página 41.

YOUNG, S. Hmms and related speech recognition technologies. In: *Springer Handbook of Speech Processing*. [S.l.]: Springer, 2008. p. 539–558. Citado na página 24.

ZUBEN, F. J. V.; ATTUX, R. R. Máquinas de vetores-suporte. Citado 2 vezes nas páginas 42 e 43.

Apêndices

APÊNDICE A – REPOSITÓRIO DO ALGORITMO DE TREINAMENTO

O link abaixo fornece o acesso ao repositório do algoritmo criado no desenvolvimento deste trabalho.

https://github.com/marioandre01/machineLearningTechniquesForVoiceRecognitionByGender_R