INSTITUTO FEDERAL DE MATO GROSSO DO SUL

CAMPUS CORUMBÁ

JOÃO FELIPE MOREIRA DE SOUZA

COMPARAÇÃO DE DESEMPENHO DOS CLASSIFICADORES NAIVE BAYES, KNN E C4.5 PARA UM CONJUNTO DE DADOS DE IMAGENS DE GESTOS DE LINGUA DE SINAIS

Corumbá- MS

2019

JOÃO FELIPE MOREIRA DE SOUZA

COMPARAÇÃO DE DESEMPENHO DOS CLASSIFICADORES NAIVE BAYES, KNN E C4.5 PARA UM CONJUNTO DE DADOS DE IMAGENS DE GESTOS DE LINGUA DE SINAIS

Trabalho de Conclusão do Curso apresentado ao curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Mato Grosso do Sul – Campus Corumbá, como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas

Orientador: Prof. Msc. Diego Saqui

Coorientador: Prof. Esp. Frank Castilio Pinheiro de Alencar

Corumbá -MS

2019

“Se A é o sucesso, então A é igual a X mais Y mais Z. O trabalho é X; Y é o lazer; e Z é manter a boca fechada. “

Albert Einstein

Resumo

A Língua Brasileira de Sinais (LIBRAS) é usada por surdos dos centros urbanos brasileiros. Segundo a Lei Nº 10.436, de 24 de Abril de 2002, LIBRAS é “a forma de comunicação e expressão em que o sistema linguístico de natureza visual-motora, com estrutura gramatical própria, constitui um sistema linguístico de transmissão de ideias e fatos, oriundos de comunidades de pessoas surdas do Brasil”. Ao longo dos anos, tecnologias e pesquisas têm surgido para ajudar na comunicação com os ouvintes. O reconhecimento de gestos por Visão Computacional (VC) é um eixo tecnológico e linha de pesquisa potencial e em desenvolvimento. Embora existam avanços, traduzir LIBRAS para língua portuguesa é um desafio a ser muito pesquisado, principalmente, no estudo de métodos classificadores capazes de determinar com exatidão os possíveis gestos. Portanto, neste estudo é apresentado uma comparação dos algoritmos Naive Bayes, K-*Nearest Neighbors* (KNN) e C4.5, para classificar gestos de línguas de sinais. Os algoritmos foram aplicados para características (dados) obtidos de 3143 imagens abordando letras de um alfabeto de língua de gestos. Para os testes realizados, uma ferramenta para comparação dos algoritmos foi desenvolvida na linguagem Python, permitindo avaliar a acurácia geral e a aplicação do teste estatístico T-pareado, que permite comparar o resultado médio de dois algoritmos classificadores e determinar se esses apresentaram resultados equivalentes ou se um é melhor. As taxas de acerto (acurácia geral) obtidas por validação cruzada com 10 *folds* foram de 91.13% para o classificador C4.5, 90.18% com o classificador KNN ambos se distanciando do algoritmo Naive Bayes que alcançou apenas 52.33% de acerto. O teste estatístico demonstrou, comparando o C4.5 e o KNN, que os classificadores são diferentes. Com base nos resultados, o classificador C4.5 mostrou ser a melhor opção para a base de dados de línguas de sinais utilizada.

Lista de abreviaturas e siglas

AM	Aprendizado de Máquina
ASL	*American Sign Language*
CSV	*Comma Separated Values*
DCSF	*Depth Cuboid Similarity Features*
IA	Inteligência Artificial
KNN	K-Nearest Neighbors
LIBRAS	Língua Brasileira de Sinais
MAD	*Multimodal Action Data set*
NB	Naive Bayes
PDI	*Pentaho Data Integration*
RGB	Vermelho-*Red*, Verde-*Green*, Azul-*Blue*
RNA	Rede Neural Artificial
RP	Reconhecimento de Padrões
VC	Visão Computacional

Lista de Tabelas

Lista de Figuras

Sumário

1. INTRODUÇÃO

A perda auditiva se baseia em um problema sensorial não visível, onde o indivíduo apresenta dificuldades na detecção e percepção dos sons. Complicações são causadas ao desenvolvimento desse indivíduo em razão da natureza complexa do ser humano, visto que padrões sociais, emocionais, linguísticos e intelectuais estão interligados (ARAÚJO; LACERDA, 2008).

Em 24 de abril de 2002, a Lei Federal nº 10.436, reconheceu a Língua Brasileira de Sinais (LIBRAS) como língua oficial dos surdos (BRASIL, 2002), e foi regulamentada pelo Decreto nº 5.626 de 22 de dezembro de 2005 (BRASIL, 2005), mudanças importantes foram feitas em relação à comunidade de surdos. De acordo com o Art. 1º da Lei Federal, é descrito:

> *É reconhecida como meio legal de comunicação e expressão a Língua Brasileira de Sinais - Libras e outros recursos de expressão a ela associados.*
> *Parágrafo único. Entende-se como Língua Brasileira de Sinais - Libras a forma de comunicação e expressão, em que o sistema linguístico de natureza visual-motora, com estrutura gramatical própria, constitui um sistema linguístico de transmissão de ideias e fatos, oriundos de comunidades de pessoas surdas no brasil (BRASIL, 2005).*

Nesse contexto, o uso da LIBRAS passou a ser um direito dos surdos. Além disso, a definição da pessoa surda, conforme o decreto citado acima, passou a ser aquela pessoa que, apesar da perda auditiva, compreende o mundo e, ao mesmo tempo, interage com ele por meio de experiências visuais.

Apesar de haver iniciativas sociais para integrar as pessoas com a comunidade surda, tendo em vista que o número de pessoas falantes que são fluentes em LIBRAS ainda é baixo, pesquisadores têm buscado novas abordagens para o problema em campos tecnológicos, e área de Visão Computacional (VC) tem sido um deles.

VC pode ser entendida como a ciência que estuda e desenvolve tecnologias que permitem que máquinas enxerguem e extraiam características do meio, por meio de imagens capturadas por diferentes tipos de sensores e

dispositivos (BALLARD; BROWN, 1982). Prince (2012) descreve que o objetivo da VC é extrair informações ou características que venham a ser úteis; e Shapiro e Stockman (2001) citam que seu propósito é tomar decisões a partir dessas informações sobre o ambiente e cenas por meio de imagens. Segundo Barelli (2018) “programar computadores para enxergar o ambiente ao nosso redor e torná-los capazes de reconhecer e extrair informações de objetos são tarefas complexas que necessitam de fortes conhecimentos matemáticos e de programação”. Junto à VC, o aprendizado de máquina (AM) oferece métodos para automatizar a aquisição de modelos visuais, adaptando tarefas e representação, traduzindo sinais em símbolos, construindo sistemas de processamento de imagens treináveis, concentrando a atenção no objeto alvo. A capacidade de raciocinar e aprender são as duas principais competências associadas a sistemas que utilizam AM.

Algoritmos de AM podem ser aplicados de pelo menos duas maneiras em sistemas de VC:

- Melhorar a transformação de sinais capturados em representações internas, e.
- Preencher as falhas entre as representações internas do ambiente e a representação do conhecimento necessário ao sistema para executar a sua tarefa (SEBE et al., 2005).

Um importante método aplicado em VC é a segmentação que é uma técnica de processamento de imagens que consiste em separar a área que representa o objeto de interesse na imagem, como por exemplo, a mão e o fundo. Somente com o objeto segmentado é possível obter características que tornam possível classificá-lo (BARELLI, 2018). Em um sistema de reconhecimento de gestos a classificação permite, por exemplo, diferenciar um determinado gesto de outro.

Um objeto em uma imagem ou cena de vídeo possui diversas características que podem ser usadas para classificá-los sendo alguns baseados no contorno, raio a partir do centro, entre outros que são obtidos a partir de sua forma. Essas características (ou atributos), podem ser extraídas a partir do objeto segmentado e compõem um descritor (conjunto de dados que representam a

essência de um todo ou parte da imagem) (NIXON; AGUADO, 2012). Estes descritores devem ser capazes de permitir sua classificação em relação a outros presentes em imagens.

Diversos tipos de descritores são explorados no reconhecimento de gestos na literatura e alguns exemplos são destacados a seguir. Yeo, Lee e Lim (2013) e Bastos, Angelo e Loula (2015) utilizaram descritores de forma que caracterizam a forma de objetos como bordas, Histograma de Gradientes Orientados (HOG) e Momentos Invariantes de Zernike (ZIM). Nimbalkar, Karhe e Patil (2014), usaram em seu trabalho descritores de textura (baseado em Viola-Jones, 2001) e de forma (baseado em Wavelets). Chen, Gao e Ma (2000), se basearam no descritor de entropia da informação (capturando os ruídos) em sua pesquisa. A entropia mede o quão irregular é uma região da imagem. Chang et al. (2013) propuseram um método de reconhecimento baseado na profundidade e na relação entre os dedos. Escalante, Morales e Sucar (2016) exploraram o descritor de profundidade (DCSF) no conjunto de dados MSRDaily3D que compreende 16 ações associadas às atividades diárias e a maioria das ações envolve interação homem-objeto, descritor que inclui: ângulos ósseos entre as articulações, diferenças entre articulações no conjunto de dados *Multimodal Action Data set* (MAD) que contém 35 ações diferentes contendo gestos e não gestos. Para o *data set* chamado Montalbano, tentaram reconhecer gestos culturais italianos de 20 categorias e os descritores usados combinam profundidade, RGB (Vermelho-Verde-Azul) e esqueleto por meio de redes convulacionais.

Acompanhado a esses descritores, algoritmos de AM têm sido aplicados. Juntos permitem construir sistemas baseados em Reconhecimento de Padrões (RP). Um exemplo de estrutura desse sistema de RP baseada no esquema de Theodoridis e Koutroumbas (2008) é apresentada na Figura 1.

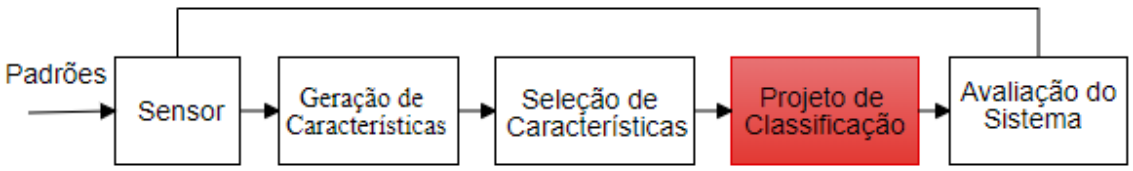

Figura 1.1 - Estrutura de sistemas de Reconhecimento de Padrões (adaptado de Theodoridis e Koutroumbas (2008)).

Na Figura 1, é possível verificar a etapa do projeto de classificação e, portanto, a necessidade de projetar e avaliar os algoritmos de classificação. Essa tarefa é explorada em reconhecimento de gestos e alguns exemplos são destacados na sequência.

Nimbalkar, Karhe e Patil (2014) exploraram o algoritmo dos K-Vizinhos mais próximos ou do inglês K-Nearest Neighbors (KNN) para o reconhecimento de gestos o algoritmo opera bem nessa finalidade. Chen, Gao e Ma (2000), utilizaram do algoritmo C4.5 para a classificação de gestos da linguagem de sinais chinesas onde obtiveram 86,2% de acurácia, e os autores pontuaram que a taxa de acerto poderia ser melhor se os dados de treinamento forem usados como dados de teste. Chang et al (2013), utilizaram árvores de decisão em tempo real para o reconhecimento de gestos gerais e gestos numéricos de língua de sinais americana (ASL) obtendo uma taxa de acerto de 95,01% e 94,33% respectivamente. Escalante, Morales e Sucar (2016) utilizaram o algoritmo Naive Bayes para classificar diferentes tipos de dados, e ressaltaram que a eficácia do classificador se compara favoravelmente com metodologias de última geração. Bastos, Angelo e Loula (2015), elaboraram um classificador baseado em Rede Neural Artificial (RNA) e utilizaram um conjunto de dados de 9600 imagens representando 40 gestos (sinais) de Libras, e a taxa de acerto foi de 96,77%.

Especialmente, em Yeo, Lee e Lim (2013), o método proposto apresentou um bom desempenho no mapeamento da mão, rastreando as mãos e os dedos utilizando um *hardware* de baixo custo, descritores e algoritmos de processamento de imagens específicos. Nos resultados, nota-se que um sistema intuitivo de Interação Homem-Computador e sistemas de jogos de movimentos podem ser alcançados com requisitos mínimos de *hardware*.

O desempenho obtido Yeo, Lee e Lim (2013) alinhado aos resultados obtidos por Nimbalkar, Karhe e Patil (2014), Chen, Gao, Ma (2000), Chang et al (2013), Escalante, Morales e Sucar (2016), sugerem que se descritores adequados forem utilizados, algoritmos clássicos de AM podem ter um bom desempenho em sistemas de reconhecimentos de gestos mesmo com equipamentos de *hardware* de baixo custo. Com base nessa perspectiva, o presente estudo avalia e compara o desempenho no quesito de acurácia geral de três algoritmos clássicos KNN, Naive Bayes e C4.5 (árvores de decisão) para

classificação de gestos de Libras baseado em características obtidas por técnicas de VC. Pretende-se no futuro, integrar um desses algoritmos a um sistema de RP para o reconhecimento de gestos de Libras.

1.1 Objetivos

1.1.1. Objetivo geral

Avaliar o desempenho dos algoritmos Naive Bayes, KNN e C4.5 (Árvores de Decisão) para classificação de gestos de Libras baseado em características obtidas por técnicas de VC.

1.1.2. Objetivos específicos

- Auxiliar no desenvolvimento de um método para melhorar o reconhecimento da mão.
- Desenvolver um aplicativo utilizando os algoritmos de classificação explorados para a análise de dados de gestos de Libras representados por características obtidas por VC.
- Escrever e publicar artigos com base no estudo realizado.

1.2 Organização do Trabalho

O trabalho encontra-se organizado da seguinte maneira:

No Capítulo 1 é apresentado a contextualização e objetivo o qual o trabalho está inserido.

O Capítulo 2 apresenta os principais conceitos de aprendizado de máquina e visão computacional em imagens, processamento de imagens e ferramentas de aprendizado que auxiliam no desenvolvimento do trabalho, além de classificadores que fazem parte dos conhecimentos necessários para o bom entendimento deste trabalho. Como as áreas de estudo são muito amplas, procurou-se direcionar o assunto para os tópicos de maior relevância.

No Capítulo 3 é feita, de uma forma geral, a descrição do método e aplicativo abordado e desenvolvido neste trabalho.

A análise dos resultados obtidos com os experimentos está descrita no Capítulo 4.

Por fim, no Capítulo 5 é apresentado as considerações finais sobre o desenvolvimento do trabalho, suas contribuições e sugestões para trabalhos futuros.

2. Referencial Teórico

2.1 Libras

Língua Brasileira de Sinais (LIBRAS) é a língua de sinais usada por surdos dos centros urbanos brasileiros. De acordo com a Lei Nº 10.436, de 24 de Abril de 2002, pode-se dizer que a LIBRAS é “a forma de comunicação e expressão, em que o sistema linguístico de natureza visual-motora, com estrutura gramatical própria, constitui um sistema linguístico de transmissão de ideias e fatos, oriundos de comunidades de pessoas surdas do Brasil”. Não somente em LIBRAS, mas diversos outros gestos humanos são um meio de interação não verbal. Esses gestos são caracterizados por ações simples como apontar e mover objetos até movimentos mais complexos que expressam os nossos sentidos e nos permitem nos comunicar (PAVLOVIC, 1997). A utilização de recursos para diminuir as dificuldades e limitações independente da deficiência é uma prática que deve ser empregada nas escolas, maximizando o desenvolvimento educacional respeitando as suas necessidades e limitações (BAQUETA; BOSCARIOLI, 2011). Além das escolas, esses recursos se fazem necessários em empresas, instituições privadas e em quaisquer outros estabelecimentos públicos ou privados que a comunidade surda tenha acesso.

O uso das tecnologias se refere a uma nova dimensão do “saber fazer”, visto que são acessíveis a comunicação visual. Novas tecnologias possibilitam uma maior acessibilidade visual, a comunidade surda a recebe como uma potencialidade na comunicação o que estabelece novas possibilidades para o seu processo educacional (LOPES, 2017).

Há vários *softwares* com o objetivo de auxiliar a comunicação entre surdos e ouvintes como, por exemplo, o ProDeaf que possui um intérprete virtual (*plug-in*) utilizado para traduzir sites do português para LIBRAS. Apesar do aplicativo ProDeaf auxiliar o surdo para o acesso a um conteúdo, ele não é capaz de traduzir para português os sinais de LIBRAS. Para isso é necessário existir a captura e interpretação dos gestos realizados por uma pessoa, como por exemplo, por meio do uso de câmeras digitais.

2.2 Visão Computacional e Reconhecimento de Padrões

A visão computacional (VC) é uma área interdisciplinar que trata da forma como os computadores podem interpretar informações a partir de imagens ou vídeos digitais (BALLARD; BROWN; 1982). Segundo Prince (2012) o objetivo da VC é extrair informações ou características que venham a ser úteis, e segundo Shapiro e Stockman (2001) o seu propósito é tomar decisões a partir dessas informações sobre o ambiente e cenas por meio de imagens. Essa área cresce em ritmo acelerado, devido a sua importância em compreender computacionalmente o mundo que observamos em imagens.

No processamento de imagens, a análise de uma imagem envolve além da informação presente, o conhecimento armazenado anteriormente. Para que as máquinas tenham desempenho semelhante aos humanos na extração de características, é preciso que se utilize algoritmos que façam a integração de diversos dados, combinando assim o alto desempenho e baixo custo computacional. O estudo de VC envolve técnicas de obtenção de imagens, segmentação, processamento de imagens e reconhecimento de padrões.

Segundo Gonzalez e Woods (1993), segmentação é o processo de subdividir uma imagem em partes ou objetos constituintes. Gonzalez e Woods (2008) descrevem que classes geradas por algoritmos de classificação, podem representar segmentos da imagem, ou seja, podem ser usadas para segmentação. Em imagens, o objetivo desses algoritmos é classificar automaticamente todos os pixels em classes (LILLESAND, KIEFER, CHIPMAN; 2004). A segmentação é aplicada para obter o objeto alvo. A imagem passa por algoritmos que empregam técnicas que visam extrair o objeto desejado do resto da imagem. A partir da imagem segmentada é possível extrair descritores que serão usados no treinamento e na classificação do algoritmo. Esses descritores são características da imagem como textura, curvatura, entre outros.

Dentre as técnicas de segmentação, temos por movimento e por cor, que podem ser utilizadas a fim de melhorar a segmentação das imagens. A segmentação por movimento implica em subtrair a imagem do *frame* atual com uma anterior para encontrar a diferença entre as duas. Isso é feito para encontrar objetos que se movimentam, no caso, assumimos que a mão é o único objeto da

imagem se movendo. Após isso, a imagem é transformada em uma imagem binária, preto e branco, e ela conterá apenas os objetos que que se moveram de um frame para outro. A segmentação por cor basicamente identifica os objetos que possuem determinada cor e segmenta da imagem. Yeo, Lee e Lim (2013) e Basilio, et. al (2011), propõe converter a imagem, que normalmente está no padrão de cor RGB, para o padrão de cor YCbCr. Esse padrão, segundo os autores, captura com mais eficiência os tons de pele humana. Eles também definem parâmetros que representam a cor da pele humana.

Reconhecimento de Padrões (RP) é uma disciplina científica onde o principal objetivo é a classificação de objetos, além de também abranger a tarefa de agrupamento (THEODORIDIS; KOUTROUMBAS, 2008). RP considera uma série de etapas para conseguir executar a classificação:

- aquisição de dados: captura de dados por algum sensor;
- geração de características: transformação dos dados adquiridos pelo sensor em características úteis para o sistema de RP;
- seleção de características relevantes: consiste em determinar as características que serão utilizadas na etapa de classificação;
- projeto do classificador: onde deve-se observar os tipos dos dados que serão classificados e se um determinado classificador opera bem com esses dados;
- avaliação do sistema: consiste em como avaliar o sistema de classificação.

Em especial o projeto do classificador faz parte da área de aprendizagem de máquina, mais especificamente aprendizado supervisionado que é explicado no subcapítulo posterior.

2.3 Aprendizado de Máquina

Junto à VC, o aprendizado de máquina (AM) oferece métodos para automatizar a aquisição de modelos visuais, adaptando tarefas e representação, traduzindo sinais em símbolos, construindo sistemas de processamento de imagens treináveis, concentrando a atenção no objeto alvo. A capacidade de raciocinar e aprender são as duas principais competências associadas a

sistemas que utilizam AM. De acordo com Coppin (2010), o aprendizado de máquina está diretamente ligado com a inteligência, pois se um sistema realmente é capaz de aprender a exercer determinada tarefa mereça então ser chamado de inteligente.

Segundo Mitchell (1997), pode-se dizer que um programa de computador aprende a partir de uma experiência E com respeito a uma classe de tarefas T e com medida de desempenho P, se seu desempenho P na tarefa T melhora com a experiência E, pode-se notar nessa definição que existem três componentes básicos que precisam ser pensados no momento de se trabalhar com a ideia de aprendizado de máquina. O primeiro componente é a tarefa a ser realizada, quando se trabalha com mineração de dados três das tarefas mais tradicionalmente utilizadas ou realizadas são: Classificação, agrupamento e a extração de regras de associação. O segundo componente a ser trabalhado é a fonte de experiência (banco de dados). O terceiro e último componente é a medida de desempenho, é preciso ter uma tarefa para ser realizada, é preciso saber de onde se busca experiência para melhorar o desempenho daquela tarefa e é preciso de uma medida que consiga mostrar se o desempenho está melhorando ou piorando conforme esteja usando a experiência disponível.

No estudo do AM existem dois tipos clássicos de aprendizado: Aprendizado supervisionado e Aprendizado não supervisionado.

2.3.1 Aprendizado supervisionado

Aprendizado supervisionado é aquele no qual o sistema ou o algoritmo é treinado para reconhecer padrões a partir de objetos já conhecidos que podem ser determinados por humanos. Os humanos apresentam dados já classificados para o algoritmo ser estruturado para classificar novos dados. Um conjunto de características de objetos e suas respectivas classes são apresentados ao classificador. Tendo esses dados, os algoritmos supervisionados são treinados para identificar automaticamente os padrões nessas informações, assim sendo capaz de classificar novos objetos como pode ser observado na Figura 2.1.

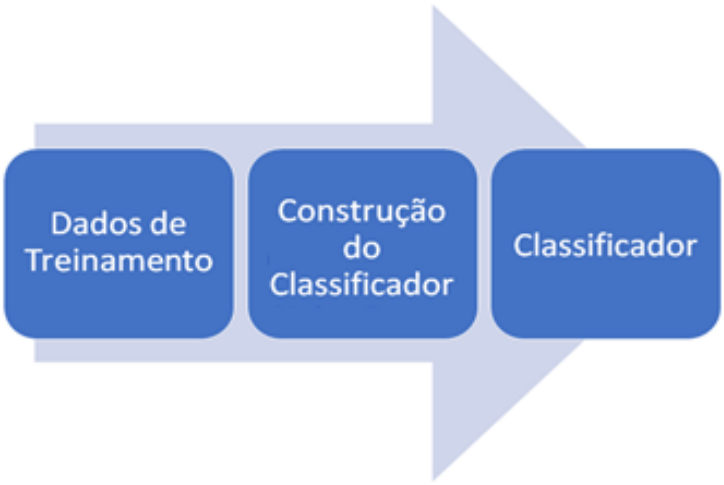

Figura 2.1 - Exemplo de construção do algoritmo de aprendizado supervisionado.

A Tabela 2.1 ilustra um exemplo de uma base de dados utilizada no processo de aprendizagem supervisionada. Nessa tabela é possível observar que as características (Dia, Aspecto, Temperatura, Umidade e Vento) estão especificadas e a saída (Decisão) também já está pré-definida. Assim o classificador é capaz, a partir dessas características, classificar se a pessoa irá jogar tênis ou não.

Tabela 2.1 - Exemplo de uma base de dados para Aprendizado Supervisionado

Dia	Aspecto	Temperatura	Umidade	Vento	Decisão
1	Sol	Quente	Alta	Fraco	Não
2	Sol	Quente	Alta	Forte	Não
3	Nublado	Quente	Alta	Fraco	Sim
4	Chuva	Agradável	Alta	Fraco	Sim
5	Chuva	Fria	Normal	Fraco	Sim
6	Chuva	Fria	Normal	Forte	Não
7	Nublado	Fria	Normal	Forte	Sim
8	Sol	Agradável	Alta	Fraco	Não
9	Sol	Fria	Normal	Fraco	Sim
10	Chuva	Agradável	Normal	Fraco	Sim
11	Sol	Agradável	Normal	Forte	Sim
12	Nublado	Agradável	Alta	Forte	Sim
13	Nublado	Quente	Normal	Fraco	Sim
14	Chuva	Agradável	Alta	Forte	Não

2.3.1.1 Naive Bayes

O classificador Naive Bayes (NB) é um algoritmo de AM indutivo (DUDA; HART, 1973) e provavelmente é o classificador mais utilizado em AM. Mitchel (1997), descreve que esse algoritmo tem um desempenho igual e muitas vezes melhor que outros algoritmos da literatura dependendo da base de dados utilizada. Ele gera classificadores probabilísticos utilizando a Teoria de Bayes, onde combina conhecimentos anteriores de uma determinada classe através de evidências selecionadas no conjunto de dados (JOHN; LANGLEY, 1995). Naive Bayes baseia-se na hipótese de independência condicional entre os atributos dada a classe, ou seja, todos os atributos são independentes entre si (ZHANG, 2004).

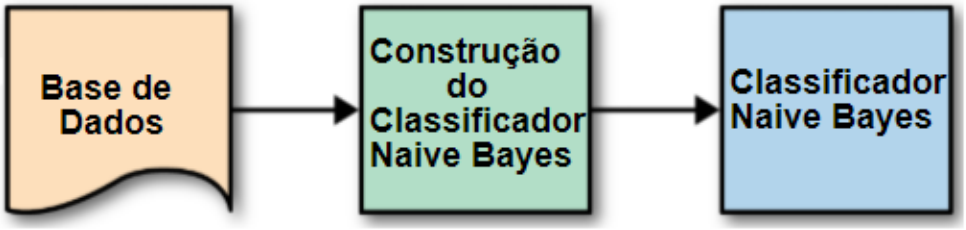

Figura 2.2 - Construção do classificador Naive Bayes.

A Figura 2.2, ilustra as fases para a construção do classificador Naive Bayes. Possuindo os dados de treinamento do objeto, é passado para a construção do classificador Naive Bayes, onde o algoritmo é treinado, para que no fim, o Naive Bayes esteja pronto para a classificação do objeto. Ele fornece uma abordagem quantitativa para pesar as evidências que suportam hipóteses. Esse raciocínio, fornece a base para o aprendizado de algoritmos, que manipulam probabilidades diretamente (MITCHELL, 1997).

O NB é um classificador probabilístico baseado no Teorema de Bayes, que mostra como determinar a probabilidade de um evento condicional através da probabilidade inversa. Para facilitar a computação, este classificador assume que a presença (ou ausência) de um atributo não tem relação alguma com qualquer outro atributo (por isso o nome Naive), ou seja, são independentes.

Partindo do teorema de Bayes, P: probabilidade; c: é a classe; X: vetor de dados; $P(c|X)$: probabilidade da classe c dado o vetor X, é chamada de probabilidade a posteriori de c; $P(c|X)$: probabilidade do vetor X dada a classe c, é chamado de probabilidade condicional (ou *likelihood*) de X; $P(c)$: probabilidade a priori da classe c; $P(X)$: probabilidade a priori do vetor de treinamento X, é uma constante independente de c.

$$P(c|X) = \frac{P(X|c) * P(c)}{P(X)} \quad (1)$$

O teorema de Bayes nos permite calcular a probabilidade a posteriori para chegar o mais perto possível da priori. A priori é a probabilidade dada sem conhecimento de qualquer outro evento, seria a classificação perfeita. A posteriori é a probabilidade condicional que é atribuída quando um evento

relevante é considerado para uma determinada classe.

$$Prob\ Posteriori = \frac{Prob\ Priori\ * Dist\ Prob}{Evidência} \quad (2)$$

A partir de procedimentos matemáticos, a equação (2) pode ser transformada na equação (3), que é utilizada na classificação do Naive Bayes.

$$H_{ML} = \underset{c \in H}{argmax} P(D|h) \quad (3)$$

A equação (3) traz a Hipótese de Máxima Verossimilhança (H_{ML}) onde, calculando a posteriori de todas as hipóteses $h \in H$, é possível encontrar a hipótese (h) mais provável – MAP (Maximum a Posteriori) – Hipótese de Máxima a Posteriori.

Quando o conhecimento a priori é equiprovável, então, $h_{ML} = h_{MAP}$, nem sempre a hipótese de máxima probabilidade a posteriori (h_{map}) traz a melhor classificação.

O classificador NB tem como vantagem o seu treinamento rápido (varredura única), rapidez para classificar, não sensível a características irrelevantes, lida com dados reais e discretos e lida bem com dados contínuos. Já a sua desvantagem é que assume que as características são independentes.

2.3.1.2 KNN

O algoritmo *K-Nearest Neighbors* (KNN), também conhecido como K-Vizinhos Mais Próximos, é um dos algoritmos de classificação mais simples de ser compreendido. Esse classificador opera por meio do método de aprendizagem supervisionada, logo é necessário que sejam apresentados previamente, dados rotulados por especialistas para o processo de treinamento.

No exemplo da Figura 2.3 é considerado o problema de classificação de peças baseado nas características de massa e área de cada uma delas.

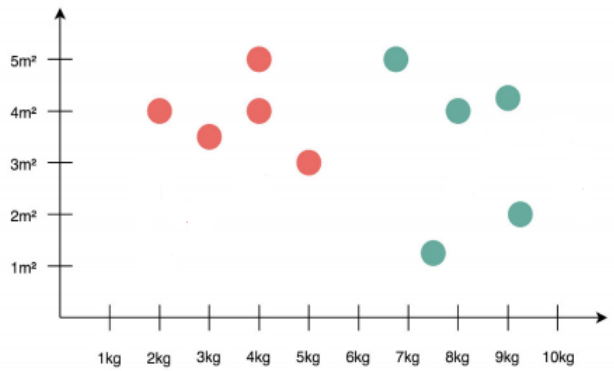

Figura 2.3 - Gráfico dos objetos e suas características.

Na Figura 2.3, é possível observar as características de massa e área de cada objeto em um espaço euclidiano bidimensional. Os objetos definidos à esquerda, na cor vermelha, indicam a classe A; os definidos à direita, na cor verde, indicam uma classe B. É importante lembrar que essas indicações das características desse objeto, tanto quanto as suas classes são informadas na base de dados para que assim o algoritmo possa treinar e aprender o que aqueles objetos são para assim classificar novos objetos.

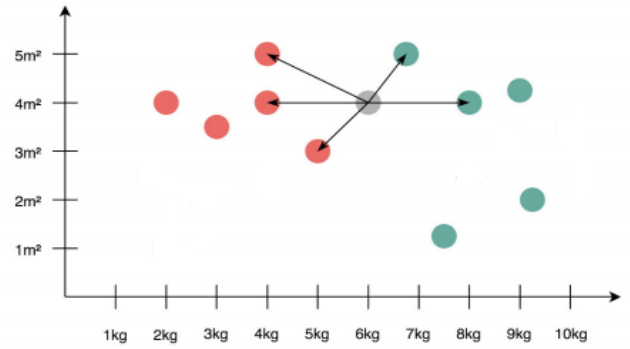

Figura 2.4 - Gráfico demonstrando uma situação onde K=5.

Para classifica-lo, o KNN calcula a distância euclidiana entre o objeto e os seus K vizinhos mais próximos. Considerando, por exemplo, K=5, a distância entre os vizinhos mais próximos será verificada. A Figura 2.4 ilustra esse procedimento na etapa de classificação. Dos 5 objetos mais próximos ao objeto não classificado representado na cor cinza, três pertencem à classe vermelha e dois à verde, logo, o algoritmo K-NN define o objeto desconhecido como pertencente à classe vermelha.

Organizando uma tabela com atributos e classificação de cada atributo a fim de calcular a distância euclidiana para com o objeto de interesse temos:

Tabela 2.2 - Tabela com os atributos e classificação de cada objeto.

#	kg	m²	Classe
1	4	2	A
2	3	3.5	A
3	4	4	A
4	4	5	A
5	5	3	A
6	6.5	5	B
7	7.5	1	B
8	8	4	B
9	9	4.5	B
10	9.5	2	B

A partir da Tabela 2.2, que traz os valores dos atributos e sua classificação, podemos utilizar a função da distância euclidiana para estabelecer quais objetos estão mais próximos do objeto de interesse para determinar a classe do objeto. A distância euclidiana é calculada a partir da equação 4, onde d_E: é a distância euclidiana; x, y: são os objetos; $x_i \in y_i$: os atributos dos objetos.

$$d_E(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4}$$

Onde se procura calcular a distância do objeto x ao y, para isso, usamos os valores dos seus atributos, nesse exemplo kg e m^2. Na equação 5 o registro que queremos classificar tem valores 6 e 4. Logo:

$$d_E(x,y) = \sqrt{(kg(x) - kg(y))^2 + (m^2(x) - m^2(y))^2} \quad (5)$$

$$d_E(1,11) = \sqrt{(2-6)^2 + (4-4)^2}$$

$$d_E(1,11) = \sqrt{(-4)^2 + (0)^2}$$

$$d_E(1,11) = \sqrt{16+0}$$

$$d_E(1,11) = 4$$

Realizando o cálculo da função para todos os objetos como pode ser observado na Tabela 2.3, pode se observar que para k=5 o objeto que era desconhecido passa a pertencer à classe A.

Tabela 2.3 - Tabela com os valores da distância euclidiana e a classe do objeto desconhecido

#	kg	m²	Classe	Distância para a instância #11
1	2	4	A	4
2	3	3.5	A	3.04
3	4	4	A	2
4	4	5	A	2.23
5	5	3	A	1.41
6	6.5	5	B	1.11
7	7.5	1	B	3.35
8	8	4	B	2
9	9	4.5	B	3.08
10	9.5	2	B	4.03
11	6	4	A	

O KNN tem dificuldade em estimar o valor de K pois, se o K for muito pequeno, a classificação fica sensível a pontos de ruído, se o K for muito grande,

a vizinhança poderá incluir elementos de outras classes, além disso, é preciso sempre escolher um valor ímpar para K, assim evitando empates na votação.

O KNN tem como vantagem sua técnica simples e de fácil implementação, sua flexibilidade e em alguns casos apresente ótimos resultados. Já as desvantagens do KNN são: a precisão da classificação pode ser muito degradada pela presença de ruído ou características irrelevantes, pode consumir muito tempo quando o conjunto de treinamento é muito grande e classificar um objeto desconhecido pode ser um processo muito completo, pois precisa de um cálculo de distância para cada treinamento.

2.3.1.3 C4.5

Árvores de decisão, são estruturas de dados que são formadas a partir do agrupamento de elementos que armazenam informações. As árvores possuem um nó raiz, podendo ser algum descritor, que estabelece um nível hierárquico e ligações para outros elementos, que são chamados de filhos. Esses filhos podem ter seus próprios filhos, quando o mesmo não possui filho, é conhecido como folha terminal (CAMPOS, 2017).

O C4.5 é um dos algoritmos mais utilizados na literatura, por ter mostrado ótimos resultados em problemas de classificação, e tem como objetivo gerar um classificador na forma de uma árvore de decisão. O algoritmo C4.5 (QUINLAN, 1993) é uma evolução do ID3 (QUINLAN, 1986). As principais mudanças em relação ao ID3 são:

- O algoritmo C4.5 define um limiar e divide os exemplos de forma binária;
- O algoritmo C4.5 permite que os valores desconhecidos para uma determinada características sejam representados como “?”, e o algoritmo trata essas características de forma especial;
- Utiliza a medida de razão de ganho para selecionar a característica que melhor divide os exemplos.

- O algoritmo C4.5 faz a busca na árvore, de baixo para cima, e transforma em nós folha aquelas características que não apresentam nenhum ganho significativo.

O C4.5 utiliza a Entropia, que é uma medida definida na teoria da informação. A entropia define a pureza de um conjunto de instâncias, e sua fórmula pode ser vista na equação 6.

$$Entropia(S) = -(p^{+} * log_2\, p^{+}) - (p^{-} *\ log_2\, p^{-}) \qquad (6)$$

Na equação de Entropia temos p^{+} que equivale a proporção de instâncias positivas, p^{-} corresponde às proporções de instâncias negativas e S corresponde à um conjunto de instâncias.

Após o cálculo da Entropia de todas as características, uma estrutura de árvore é formada, como pode ser visto na Figura 2.5, a característica de maior Entropia é colocada no topo da árvore.

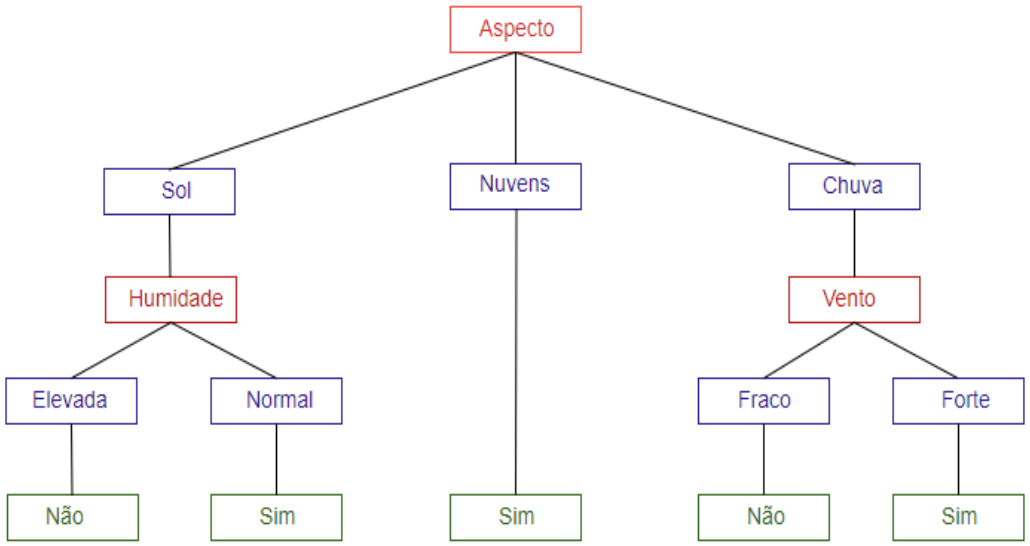

Figura 2.5 - Exemplo jogar tênis num modelo de árvore de decisão.

O C4.5 atua em duas linhas de atuação para minimizar o *overfitting* (é um termo usado em estatística para descrever quando o modelo se mostra ineficaz para prever novos resultados), em aprendizado de árvores de decisão:

- Parar de expandir um ramo antes de atingir a classificação perfeita (Entropia = zero);

- Podar a árvore construída sem a preocupação de se evitar *overfitting*.

A vantagem do classificador C4.5 é que ele realiza uma busca de baixo para cima, transformando em nós folhas aqueles nós que não representam nenhum ganho significativo, leva em consideração as características de mais relevância e combate o problema de *overfitting*. Já a sua desvantagem é que quando a árvore fica muito complexa a sua compreensão fica comprometida, a saída seria criar uma regra para cada classe, porém para isso seria necessário quantidades consideráveis de CPU e memória para construí-las.

2.3.2 Aprendizado não supervisionado

O uso de métodos de aprendizagem não supervisionados para a construção de extratores de características, tem uma longa e bem-sucedida história em reconhecimento de padrões e visão computacional.

O aprendizado não supervisionado, permite abordar problemas com pouca ou nenhuma ideia do resultado, ou seja, os dados apresentados ao algoritmo não supervisionado não são rotulados, as variáveis são fornecidas sem informar a classe de saída correspondente. Na Figura 2.6, é apresentado como se dá a identificação de novos objetos na aprendizagem não supervisionada.

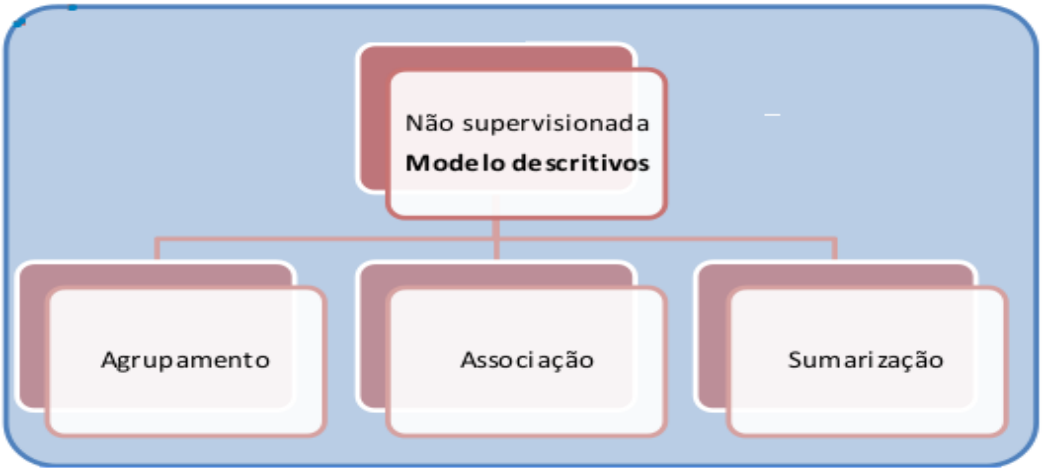

Figura 2.6 - Demonstração dos Modelos Descritivos.

Na aprendizagem não supervisionada, os algoritmos são deixados a si próprios para descobrir automaticamente estruturas interessantes nos dados como padrões, perfis, itens semelhantes e relações em um conjunto de dados. Em geral, é utilizado para encontrar aglomerados de conjuntos de dados semelhantes entre si (*clusters*). Essas característica dos métodos não supervisionados, podem ser vistas nas Figuras 2.7 e 2.8.

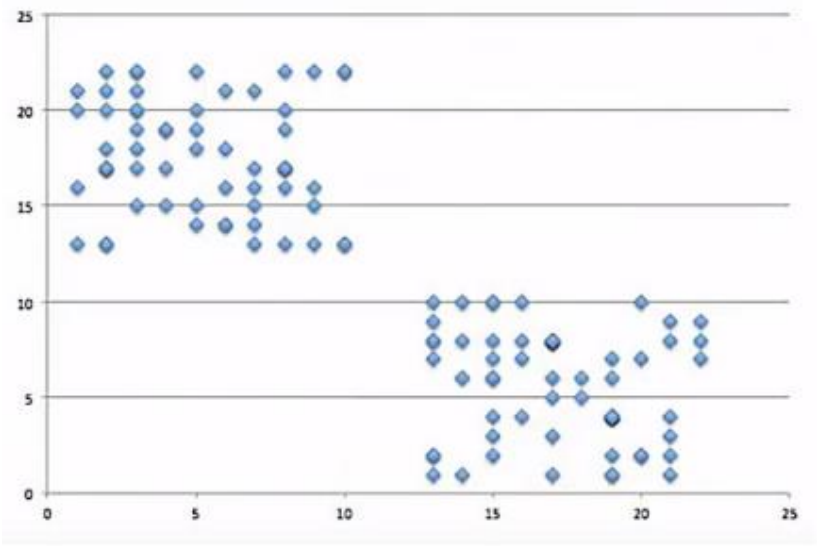

Figura 2.7 - Objetos disponibilizados sem informar a classe.

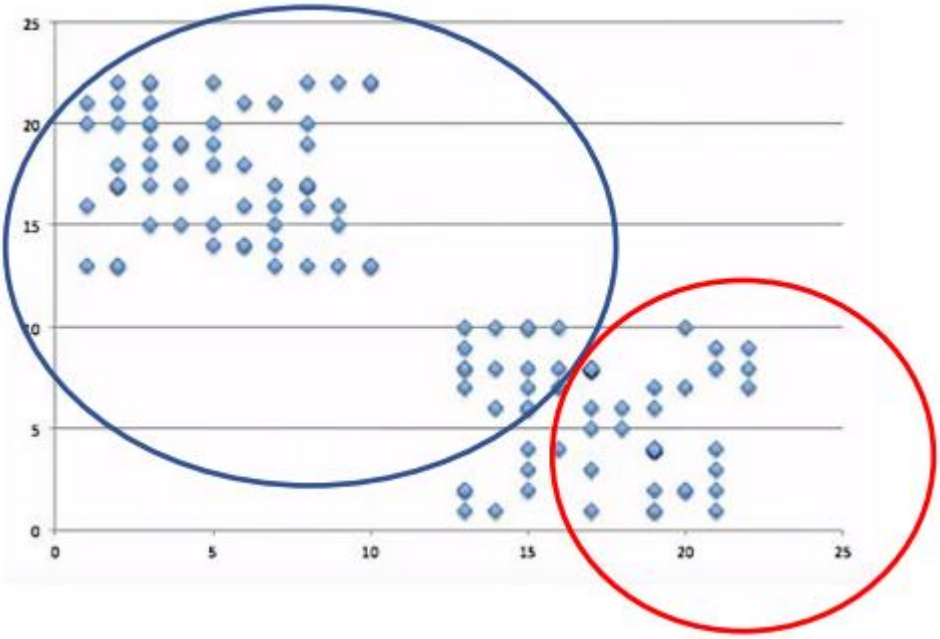

Figura 2.8 - Algoritmo agrupando os dados.

Pode se notar na Figura 2.9 que, após os dados serem assimilados e agrupados de acordo com suas semelhanças e padrões, o algoritmo consegue separá-los e dizer qual classe esse objeto pertence.

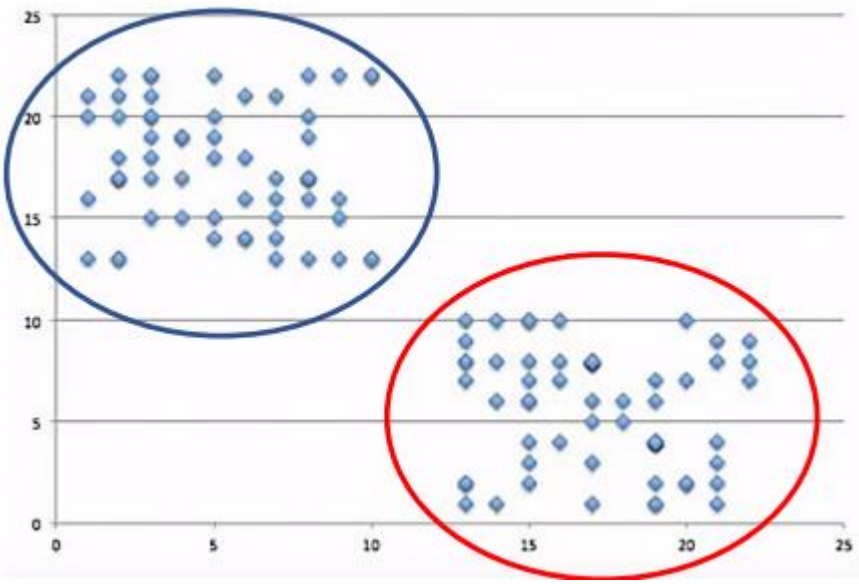

Figura 2.9 - Objetos e suas classes definidas.

O aprendizado não supervisionado pode ser usado para aprender recursos invariantes através de algoritmos de agrupamento. Um algoritmo de aprendizado não supervisionado muito conhecido é o K-Means (também chamado de K-Médias), é um método de *Clustering* e tem como ideia principal fornecer uma classificação de informações de acordo com os próprios dados.

2.3.3 Avaliação e Testes de Classificadores

2.3.3.1 *Holdout*

A técnica de teste *holdout*, consiste em separar em quantidades percentuais, uma parte do conjunto de dados para treinamento e outra para testes. Nessa técnica executada uma só vez o treinamento dos dados para construção do modelo, que depois é classificado e testado com a parte separada.

2.3.3.2 Validação Cruzada (*Cross Validation*)

A validação cruzada (*Cross Validation*), consiste em escolher o número de grupos que serão organizados em treinamentos e testes. É um *loop*, onde o usuário indica o número de *folds*, ou seja, quantas vezes o código executará aquele treinamento para classificar os dados. Na Figura 2.10, pode ser visto um exemplo de interação onde o número definido pelo usuário de k-folds=5.

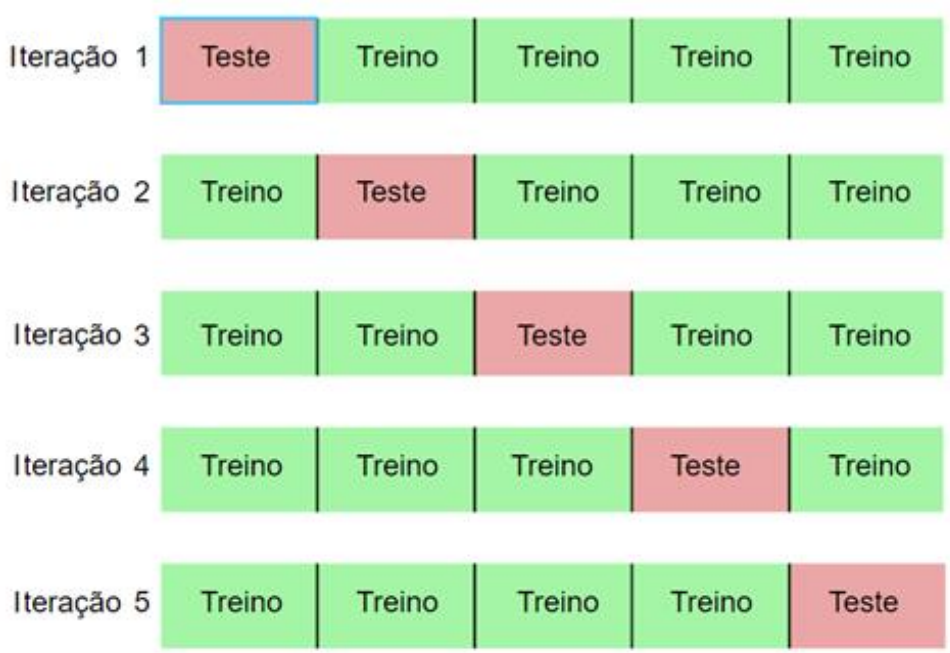

Figura 2.10 - Interação Cross Validation para k-folds = 5.

2.4 Ferramentas de Visão Computacional e Aprendizado de Máquina

2.4.1 Linguagens de Programação

Implementar algoritmos de Inteligência Artificial (IA) e *Pentaho Data Integration* (PDI) é complexo, porém linguagens atuais já possuem estruturas e bibliotecas que facilitam essa implementação. Para isso, é necessário ter uma boa linguagem de programação, com um bom desempenho em tempo de execução, bom suporte a ferramentas e com uma grande e ativa comunidade de

programadores. Algumas das possíveis linguagem que podem ser utilizadas para essa finalidade são descritas a seguir.

2.4.1.1 Python

É uma linguagem de programação mantida pela organização Python Software Foundation sem fins lucrativos. Python, é aplicada para o desenvolvimento de sites e desktops, no ensino, aplicações com números e principalmente aplicações científicas podendo ser desenvolvidas de forma orientada a objetos ou estruturada (PSF, 2019).

2.4.1.2 C++

É uma extensão da linguagem de programação C originada da linguagem B escrita por Ken Thompson. C foi desenvolvida inicialmente nos laboratórios de Bell da American Telephone and Telegraph (AT & T) entre 1969 e 1973. Bjarne Stroustrup[3], apresentou ao mundo seu projeto, uma linguagem forte e capaz de resolver problemas complexos com seu vasto leque em recursos. Com o passar do tempo C++ foi chamado, segundo Stroustrup, de “C com classes”; classes essas que formam peça chave na programação orientada a objetos.

2.4.1.3 Java

É uma linguagem de programação orientada a objetos e plataforma computacional lançada pela Sun Microsystems em 1995. Em 2008 o Java foi adquirido pela empresa Oracle Corporation. Java também é uma tecnologia usada para desenvolver aplicações, executar jogos e usar diversos serviços (JAVA, 2019).

2.4.2 Bibliotecas

Assim como nas Linguagens de Programação, bibliotecas para a implementação e desenvolvimento de sistemas voltados à área de Inteligência Artificial (IA) e *Pentaho Data Integration* (PDI), se fazem necessárias pois traz recursos importantes que auxiliam na criação desses sistemas. Algumas dessas bibliotecas são apresentadas a seguir.

2.4.2.1 OpenCV

OpenCV, que é uma biblioteca multiplataforma, projetada para o desenvolvimento de aplicativos na área de Visão Computacional, Processamento de Imagens, Estrutura de Dados e Álgebra Linear. Livre para o uso acadêmico e comercial. Essa biblioteca tem um conjunto de funções que torna simples a tarefa de manipular e processar imagens digitais e funciona de forma integrada com Python, C++ e Java. O *OpenCV* foi projetado para eficiência computacional e com um forte foco em aplicativos em tempo real. A biblioteca é otimizada e pode ser utilizada para trabalhar com processamento em vários núcleos (OPENCV, 2019).

2.4.2.2 Pandas

Pandas é uma biblioteca *open source,* que fornece estruturas de dados de alto desempenho e ferramentas de análise de dados fáceis de usar para a linguagem de programação Python (PANDAS, 2019).

2.4.2.3 Scikit-Learn

Scikit-Learn é uma biblioteca simples e eficiente de aprendizado de máquina que auxilia na mineração de dados e na análise de dados. É uma ferramenta de código aberto para o Python, e pode ser utilizado com algumas finalidades como na detecção de spam, reconhecimento de imagem, resposta a

medicamentos, preços de ações, agrupamento de resultados de experimentos entre outras (SCIKIT-LEARN, 2019).

2.4.3 Weka

Weka é um software de código aberto, escrito em Java, que contém uma coleção de algoritmos de aprendizado de máquina para tarefas de mineração de dados. Ele contém ferramentas para preparação de dados, classificação, regressão, agrupamento, mineração de regras de associação e visualização. O software começou a ser escrito em 1993, na Universidade de Waikato, Nova Zelândia. A nível de curiosidade, weka é um pássaro que não voa e tem uma natureza inquisitiva, e é encontrado apenas nas ilhas da Nova Zelândia (WEKA, 2019).

3. Materiais e Métodos

3.1 Conjunto de dados utilizados

Para validação dos algoritmos K-*Nearest Neighbors* (K-NN), C4.5 e Naive Bayes (NB), foram utilizadas 3143 imagens de gestos onde estavam disponíveis seus respectivos significados. As imagens utilizadas foram disponibilizadas na plataforma GitHub[1] que é reconhecida pela comunidade de pesquisadores e programadores. Alguns exemplos dessas imagens podem ser visualizados na Figura 3.1 e foram adquiridas por câmeras comuns, mas poderiam ser obtidas por qualquer dispositivo de captura.

Figura 3.1 - Exemplos das imagens obtidas na base de dados.

A partir das imagens utilizadas, as seguintes características foram extraídas utilizando um algoritmo específico desenvolvido para essa finalidade: X, Y, XMIN, YMIN, Dedos, RAIOMIN, AREACIRCMIN, AREACIRCMAX, XMAX, YMAX, HU0, HU1, HU2, HU3, HU4, HU5 e HU6. As características X, Y, XMIN, YMIN e RAIOMIN, são utilizadas para calcular a massa central do objeto, ou seja, o centro da imagem. O atributo AREACIRCMIN é a área do círculo descrito acima. XMAX e YMAX são valores usados juntos paras identificar o centro do maior círculo dentro do contorno da mão. O atributo RAIOMAX é o raio do círculo

[1]Base de dados de imagens de Libras: https://github.com/loicmarie/sign-language-alphabet-recognizer. Acesso em 01/03/2019.

descrito acima, assim como a característica AREACIRMAX que corresponde também a esse círculo, e que também pode ser encontrado na equação (7).

$$\pi * RAIOMAX \quad (7)$$

A característica DEDOS é a quantidade de dedos presentes na imagem, identificada ao calcular convexidades na imagem. Para o sistema de captura dos dados, um dedo é definido após passar pelo método chamado *convex hull* e pelo método chamado de curvatura-k.

Os atributos HU0, HU1, HU2, HU3, HU4, HU5 e HU6, são valores (momentos) que são usados para descrever, caracterizar e quantificar a forma de um objeto. Esses momentos são calculados a partir do contorno do objeto.

Essas características e suas respectivas classes foram armazenadas em um arquivo .csv, manipulável no aplicativo Excel da Microsoft e por linguagens de programação e tem os valores separados por vírgula. Esse arquivo é representado na Figura 3.2 e posteriormente utilizado durante o projeto do classificador. A Figura 3.2 representa a estrutura dos dados no arquivo .csv utilizado, onde a última coluna é a coluna de classe.

J	K	L	M	N	O	P	Q	R	S
RAIOMAX	AREACIRCMAX	HU0	HU1	HU2	HU3	HU4	HU5	HU6	Classe
29	8222.5	0	1	2	3	6	-7	6	A
29	8361.5	0	1	2	3	6	5	6	A
29	8265.5	0	1	2	3	6	4	6	A
30	8303.5	0	1	2	3	6	5	6	A
30	8242.0	0	1	2	3	6	5	6	A
30	8165.5	0	1	2	3	6	-5	6	A
29	8216.0	0	1	2	3	6	5	6	A
29	8201.0	0	1	2	3	6	4	6	A
29	8222.5	0	1	2	3	6	5	6	A
29	8149.5	0	1	2	3	8	-4	6	A
29	8269.5	0	1	2	3	6	4	6	A
29	7988.5	0	1	2	3	8	-4	6	A
29	8131.0	0	1	2	3	7	5	6	A
29	7914.5	0	1	2	3	8	-4	6	A
29	7973.0	0	1	2	3	7	-5	6	A
29	8009.0	0	1	2	3	7	5	6	A
29	7777.5	0	1	2	3	-7	-4	6	A
29	7852.0	0	1	2	3	-7	-4	6	A
29	7894.0	0	1	2	3	7	6	6	A
29	7887.0	0	1	2	3	6	5	6	A

Figura 3.2 - Arquivo .CSV de características de imagens de gestos gerado.

3.2 Ferramentas utilizadas

Esta pesquisa consiste de uma parte do projeto de Iniciação Científica (IC) com título “Estudo da utilização de técnicas de visão computacional para o reconhecimento de gestos de Libras”, onde foram utilizadas a biblioteca OpenCV e a linguagem de programação Python na versão 3.7.3. Para dar continuidade ao desenvolvimento e uma futura integração, a linguagem Python continuou a ser utilizada. Além de possibilitar facilidade na integração, diversos recursos poderosos como a biblioteca Pandas e a biblioteca scikit-learn que auxiliam na aplicação de aprendizado de máquina e mineração de dados são disponíveis nessa linguagem e foram utilizados neste estudo. Na criação da interface foi utilizada a ferramenta Qt Designer, uma ferramenta que auxilia e facilita o desenvolvimento de interfaces para ser vinculado à um código na linguagem Python.

3.3 Procedimentos

Conforme o projeto de IC citado no sub capítulo anterior, a presente pesquisa se encaixa no bloco destacado em vermelho da Figura 3.3a. Para o entendimento dessas etapas, na Figura 3.3 é apresentado dois fluxogramas, o primeiro (Figura 3.3a) onde processo de reconhecimento de gestos como um todo é apresentado e o segundo (Figura 3.3b) onde o projeto do classificador é demonstrado.

O primeiro fluxograma, na Figura 3.3a, foi realizada em conjunto ao projeto de pesquisa de IC e consiste em:

- Sensor: Capturar a imagem onde pode ser feita através de uma *webcam,* porém neste estudo, foram utilizadas imagens já capturadas conforme citadas no subcapitulo anterior;
- Geração de Características: Segmentar a imagem, ou seja, separar o objeto de interesse, a mão, dos demais elementos da imagem;
- Selecionar as características: foi feito com base em trabalhos da literatura, onde foram observados estes trabalhos e analisadas as

características que melhor poderiam contribuir para o trabalho, gerando assim um arquivo no formato csv;

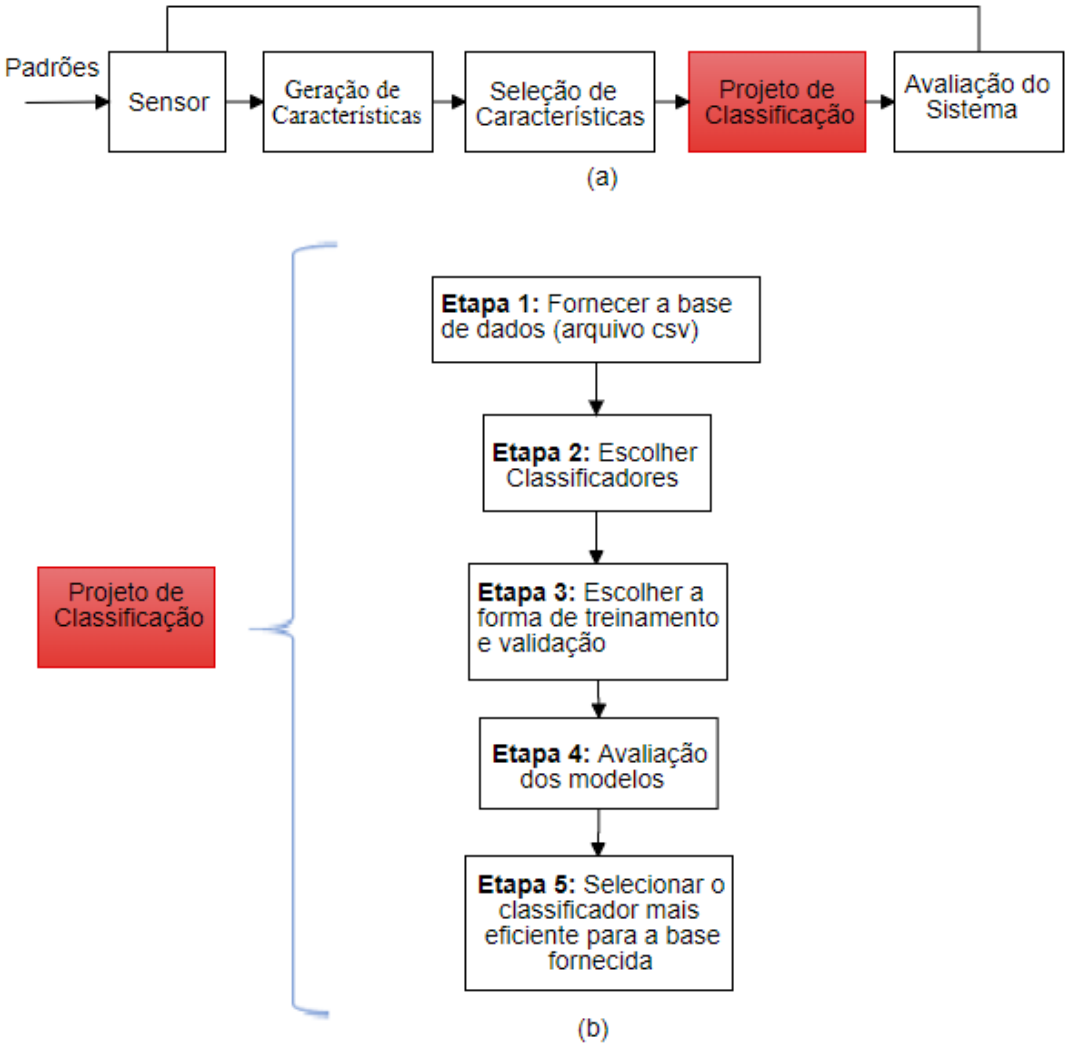

Figura 3.3 - Fluxograma do projeto de classificação.

- Projeto do Classificador: aqui que o presente estudo foca, no estudo e análise dos classificadores KNN, C4.5 e NB para o problema de linguagem de sinais, foi criado um aplicativo capaz de executar a classificação dos dados simultaneamente;
- Avaliação do sistema: a avaliação dos sistemas, dos classificadores pode ser feita com testes estatísticos afim de avaliar a qualidade dos classificadores e se eles, estatisticamente são iguais.

O segundo fluxograma, na Figura 3.3b, na vertical, consiste em selecionar determinar um classificador para usar no sistema de Reconhecimento de Padrões (RP). Para isso, na Etapa 1, é novamente utilizado o arquivo de dados .csv.

Na Etapa 2, o usuário escolhe o classificador que deseja usar para a análise de sua base de dados, sendo possível selecionar um classificador por vez ou todos simultaneamente.

Na Etapa 3, o usuário irá selecionar a forma de validação, tendo duas opções: validação cruzada (que organiza os dados em diferentes grupos) ou *houldout* (que divide a base em um percentual para treinamento e outro para testes).

A Etapa 4 consiste em classificar os dados a partir das opções selecionadas nas etapas anteriores. Neste estudo foram considerados os três algoritmos: KNN, C4.5 e NB.

A Etapa 5 consiste em utilizar a acurácia adquirida na etapa anterior para comparar os classificadores e indicar através da interface qual é o melhor classificador para a base de dados escolhida. Nessa etapa, uma opção viável que foi estruturada neste trabalho para funcionar em conjunto com a validação cruzada é a aplicação do teste estatístico T-Pareado. Esse teste permite determinar estatisticamente e com um intervalo de confiança se os resultados das execuções para os diferentes algoritmos têm médias iguais ou diferentes.

Para o procedimento no fluxograma da Figura 3.3b representado na vertical, foi desenvolvido uma aplicação, onde inicialmente foi realizado uma análise juntamente com especialista. Além disso, também foi observado o funcionamento da ferramenta Weka, que apresenta funcionalidades semelhantes a aplicação desenvolvida. Com isso, a aplicação criada seguiu o diagrama de casos de uso apresentado na Figura 3.4.

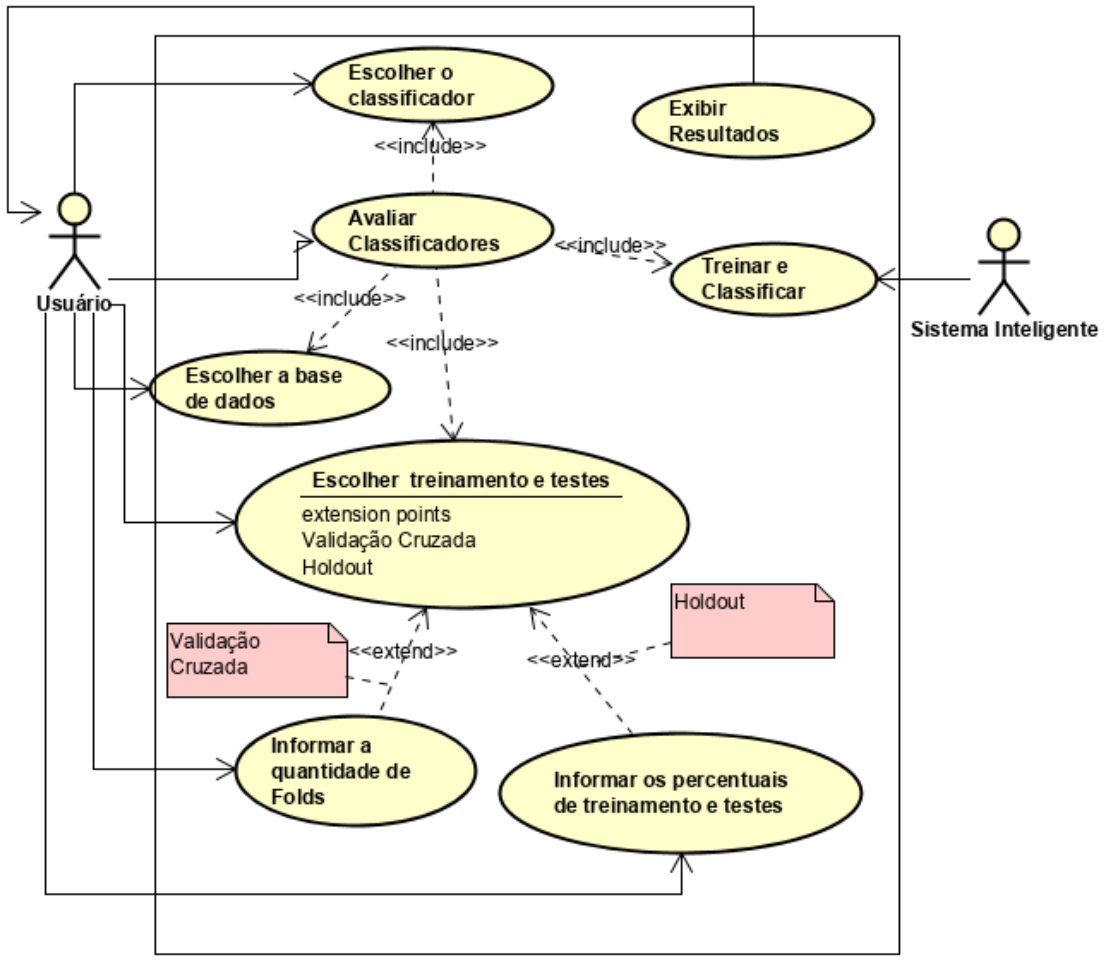

Figura 3.4 - Caso de uso do aplicativo.

Algoritmo 1 – Bibliotecas e importações

```
import sys
from PyQt5 import uic, QtGui
from PyQt5.QtWidgets import QFileDialog, QApplication, QWidget,
QLabel, QTableWidget, QTableWidgetItem, QListWidget, QListWidgetItem
import pandas as pd
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import KFold
import numpy as np
from scipy import stats

Ui_MainWindow, QtBaseClass = uic.loadUiType("interface.ui")
```

A partir do caso de uso, o desenvolvimento da aplicação foi iniciado e um trecho do código pode ser visualizado no Algoritmo 1 onde é mostrado a importação da interface criada, da biblioteca pandas e de serviços específicos utilizados da biblioteca sklearn, como os classificadores do KNN, do Naive Bayes e da Árvore de decisão.

Como pode ser observado no Algoritmo 1, foram utilizados alguns métodos da biblioteca sklearn como pode ser visto no Algoritmo 1, tais como: sklearn.neighbors import KNeighborsClassifier (algoritmo de classificação KNN), sklearn.metrics import accuracy_score (método para o cálculo da taxa de acerto dos algoritmos), sklearn.tree import DecisionTreeClassifier (algoritmo de classificação C4.5), sklearn.naive_bayes import GaussianNB (algoritmo de classificação Naive Bayes).

A interface criada para a interação do usuário, onde ele pode selecionar uma base de dados para a classificação desses dados pode ser visualizada na Figura 3.5.

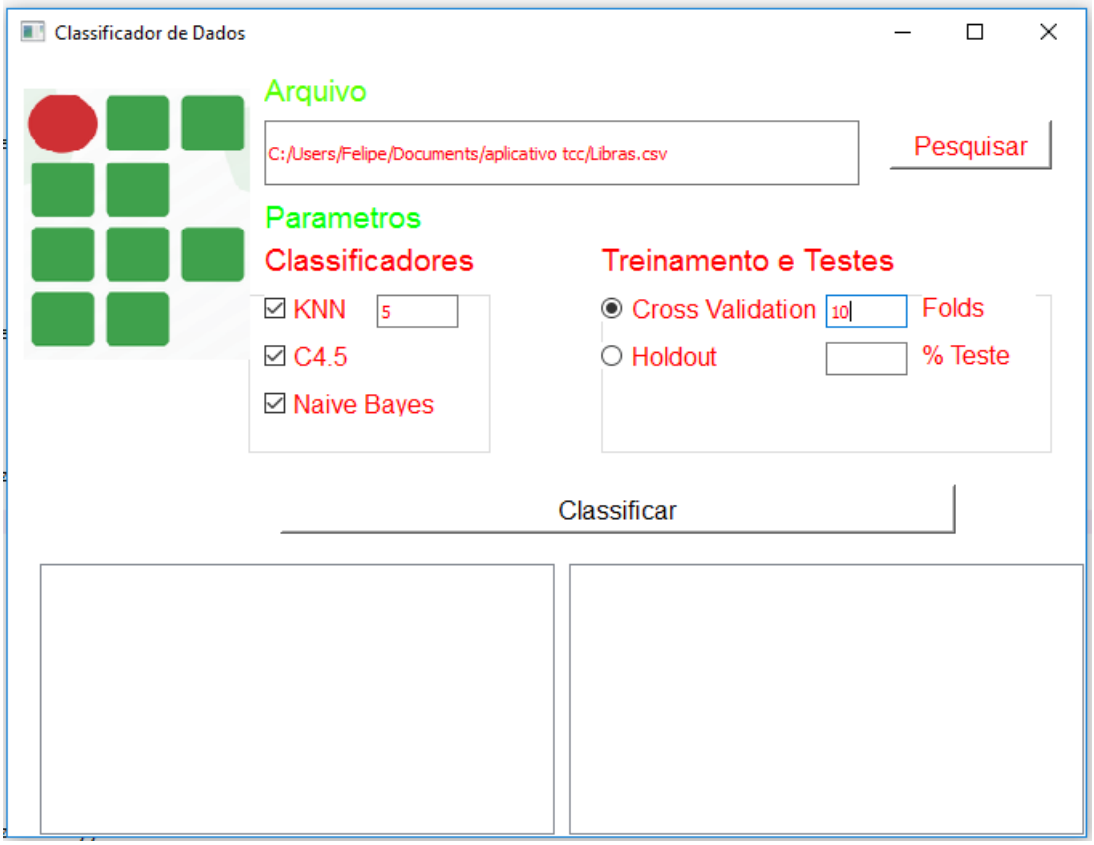

Figura 3.5 - Interface do aplicativo.

Após o usuário selecionar uma base de dados para ser classificada essa base é dividida internamente onde a última coluna representa a classe (variável y). Os outros valores dos atributos da base são armazenados na variável x.

Algoritmo 2 – Organização e padronização dos dados

```
#Carrega Dataset
dados = pd.read_csv(self.dbInput.text(), sep=';')

#Pega a coluna de classes e joga na variavel y
df = list(dados.columns.values)
colunaClasse = df[-1]
y= dados.pop(colunaClasse).values

# x = carrega as instâncias
scaler = preprocessing.StandardScaler().fit(dados.values)
x=scaler.transform(dados.values)
```

No Algoritmo 2 é apresentado um exemplo onde a base é dividida internamente em coluna de classes em y e os outros atributos em x e mostra também a padronização das variáveis. Esse procedimento ocorre após o usuário selecionar os algoritmos e o tipo de treinamento e testes.

O programa possibilita duas opções de treinamento e testes, a validação cruzada e o *holdout*. A primeira opção é a validação cruzada (do inglês *cross validation*) onde o usuário escolhe a quantidade de *folds*, ou seja, número de grupos que serão organizados em treinamento e testes. Na segunda opção, *holdout*, o usuário escolhe a porcentagem da base de dados que ele quer usar para testar.

No Algoritmo 3 são mostrados os procedimentos relacionados ao *Cross Validation*. Os *folds* são organizados a partir de x nas variáveis x_treino, y_treino, x_teste e y_teste. Os modelos para os classificadores são criados, a função fit() é utilizada para treinar os dados, a função predict() é utilizada para classificar os dados.

Algoritmo 3 – *Cross Validation*

```
for treino_index, teste_index in kf.split(x):
    x_treino, x_teste = x[treino_index], x[teste_index]
    y_treino, y_teste = y[treino_index], y[teste_index]

    #knn escolhido
    if self.cbKnn.isChecked() and self.inputK.text()!="" :
       cont_knn = 1
       #vizinhos, o usuario escolhe o valor de k
       k = int(self.inputK.text())
       classificador_KNN = KNeighborsClassifier(n_neighbors=k)
       #KNN
       #cria o modelo do KNN = treinamento
       classificador_KNN.fit(x_treino, y_treino)
       # classifica os dados do x_teste
       y_predict=classificador_KNN.predict(x_teste)
       #adiciona a acurácia na lista do Knn
       list_knn.append(accuracy_score(y_teste, y_predict,
                                              normalize=True))
    if self.cbArvores.isChecked() :
       cont_arvores = 1
       classificador_arvores = DecisionTreeClassifier(random_state=0)
       #Arvore
       #cria o modelo do Arvore = treinamento
       classificador_arvores.fit(x_treino, y_treino)
       # classifica os dados do x_teste
       y_predict=classificador_arvores.predict(x_teste)
       #adiciona a acurácia na lista do arvore
       list_arvores.append(accuracy_score(y_teste, y_predict,
                                                normalize=True))

    if self.cbNb.isChecked() :
    cont_nb = 1
    classificador_Nb = GaussianNB()
    #NB
    #cria o modelo do NB = treinamento
    classificador_Nb.fit(x_treino, y_treino)
    #classifica os dados do x_teste
    y_predict=classificador_Nb.predict(x_teste)
    #adiciona a acurácia na lista da arvore
    list_nb.append(accuracy_score(y_teste, y_predict,
                                                normalize=True))
```

No Algoritmo 4, é possível ver o funcionamento da técnica de treinamento e teste *holdout*. Uma diferença notória comparando com o *cross validation* é que, no *holdout* o processo é feito uma vez apenas diferente do Algoritmo 3 onde entra no laço de repetição for. Outra diferença é que no holtout o usuário escolher a porcentagem dos dados que deseja utilizar para realizar os testes.

Algoritmo 4 – Holdout

```
teste = int(self.inputHoldout.text()) / 100
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=teste)

#knn escolhido
if self.cbKnn.isChecked() and self.inputK.text()!="" :
     cont_knn = 1
     #vizinhos, o usuario escolhe o valor de k
     k = int(self.inputK.text())
     classificador_KNN = KNeighborsClassifier(n_neighbors=k)
     classificador_KNN.fit(x_train, y_train)
     y_predict = classificador_KNN.predict(x_test)
     list_knn.append(accuracy_score(y_test,y_predict,normalize=True))


#c4.5 escolhido
if self.cbArvores.isChecked() :
     cont_arvores = 1
     classificador_arvores = DecisionTreeClassifier(random_state=0)
     #Arvore
     #cria o modelo do Arvore = treinamento
     classificador_arvores.fit(x_train, y_train)
     # classifica os dados do X_teste
     y_predict=classificador_arvores.predict(x_test)
     #adiciona a acurácia na lista do arvore
     list_arvores.append(accuracy_score(y_test, y_predict,
     normalize=True))

#NB escolhido

if self.cbNb.isChecked() :
     cont_nb = 1
     classificador_Nb = GaussianNB()
     #NB
     #cria o modelo do NB = treinamento
     classificador_Nb.fit(x_train, y_train)
     #classifica os dados do x_teste
     y_predict=classificador_Nb.predict(x_test)
     #adiciona a acurácia na lista da arvore
     list_nb.append(accuracy_score(y_test, y_predict,
     normalize=True))
```

No Algoritmo 5 é apresentado o teste T-pareado, para o treinamento de validação cruzada com o intuito de comparar estatisticamente os classificadores e analisar se eles são iguais ou não para a classificação da base de dados usada.

Algoritmo 5 – Teste T-pareado

```
#Teste entre Knn e C4.5
if cont_knn==1 and cont_arvores==1 :
     knn_arvores =stats.ttest_rel(list_knn,list_arvores)
     pvalue1 = knn_arvores[1]

#Teste entre Knn e NB
if cont_knn==1 and cont_nb==1 :
     knn_nb =stats.ttest_rel(list_knn,list_nb)
     pvalue2 = knn_nb[1]

#Teste entre C4.5 e NB
if cont_arvores==1 and cont_nb==1 :
     arvores_nb =stats.ttest_rel(list_arvores,list_nb)
     pvalue3 = arvores_nb[1]
```

4. Experimentos e Resultados

Este capítulo aborda os experimentos realizados bem como a apresentação e a análise dos resultados obtidos.

A métrica utilizada para a avaliação é a acurácia que é obtida tanto por validação cruzada (*cross validation*) quanto por *holdout*.

O primeiro teste foi aplicado a *holdout* para treinar e classificar a base de dados de Libras. Neste primeiro experimento, foi utilizado 20% dos dados para testes e 80% dos dados para o treino. Para o algoritmo *K-Nearest Neighbors* (KNN) foi utilizado k=5. Na Tabela 4.1 é possível visualizar a acurácia geral (taxa de acerto) dos classificadores.

Tabela 4.1 - Taxa de acerto dos classificadores na técnica Holdout

Classificadores	**Acurácia**
KNN (K=5)	89.05%
C4.5	90.05%
Naive Bayes	52.68%

Na Tabela 4.1, é possível notar que o classificador C4.5 foi um pouco melhor que o algoritmo KNN e ambos se destacaram em relação ao Naive Bayes (NB) para a base de dados de linguagem de sinais utilizada.

Além dos resultados anteriores, para a separação de dados Holdout, foram realizados testes para diferentes porcentagens de dados de treinamento e testes, com o propósito de analisar a variação da acurácia dos classificadores. Essa variação de dados pode ser visualizada no gráfico da Figura 4.1.

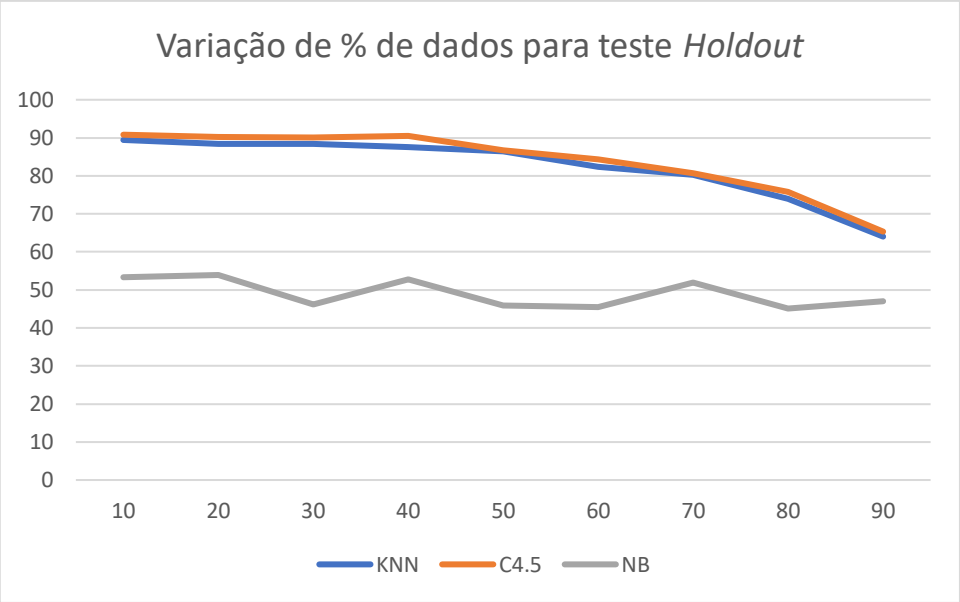

Figura 4.1 - Variação da acurácia para os dados de teste.

Nesse gráfico fica visível que na técnica *holdout* quanto menor o número de dados de treinamento, o modelo pode não ficar adequado para classificação, pois quando se aumenta os dados para testes, o número de dados de treinamento é reduzido. É importante observar que tanto para o KNN, quanto para o C4.5 que a partir dos 40%, ou seja, 60% de dados de treinamento, a curva começa a ser acentuada enquanto decresce, isso pode sugerir que nesse ponto alguns elementos (letras) da base de dados podem não ter tido amostras suficientes na fase de treinamento.

A segunda categoria de testes realizados considerou a validação cruzada. Nesse cenário, foi possível aplicar o teste estatístico t-pareado para verificar se dois resultados de classificação podem ser considerados estatisticamente iguais ou se um é melhor. Aqui foi considerado que se o p-*value* for maior que 0,05, então não é possível rejeitar a hipótese nula do teste e os resultados das médias de acurácia dos classificadores comparados são estatisticamente iguais. Se o p-*value* for menor que o liminar de 0,05, então rejeitamos a hipótese nula, e isso significa que os classificadores apresentaram médias de acurácias gerais diferentes. Com esse resultado pode-se observar que um dos dois classificadores é melhor que o outro e para determinar o melhor é necessário observar o valor da acurácia média.

No primeiro teste foram utilizados 10 *folds* e os resultados da acurácia

geral média podem ser visualizados na Tabela 4.2

Tabela 4.2 - Resultado da taxa de acerto da acurácia geral média obtida por validação cruzada com 10-folds.

Classificadores	**Acurácia**
KNN (K=5)	90.18%
C4.5	91.13%
Naive Bayes	52.33%

Na primeira coluna da Tabela 4.2 é mostrado o nome dos classificadores utilizados e na segunda coluna as acurácias desses classificadores.

O resultado do teste estatístico (p-*value*) para determinar se um classificador é melhor que o outro é mostrado na Tabela 4.3.

Tabela 4.3 - Estatística (p-value) da comparação dos classificadores.

P-VALUES		
KNN X C4.5	**KNN X NB**	**C4.5 X NB**
0.033	0.0	0.0

Na Tabela 4.3 é possível notar que o p-*value* dos classificadores KNN (k=5) e C4.5 é menor que o limiar de 0.05, logo é possível rejeitar a hipótese nula e podemos afirmar, estatisticamente, que esses classificadores apresentaram acurácia geral média diferentes. Pode-se notar que ambos quando comparados ao classificador NB, o p-*value* variou muito perto do 0, assim sendo arredondado para 0, logo, podemos afirmar estatisticamente que, o classificador NB para a base de dados é inferior em comparação ao KNN e ao C4.5.

É possível notar também que o classificador C4.5, assim como no teste de *holdout*, se saiu melhor com a base de dados de Libras, seguido pelo KNN e

ambos com uma diferença muito grande do classificador NB.

Foi realizado testes para diferentes *folds* a fim de ver a variação da acurácia dos classificadores, e isso fica visível na Figura 4.2.

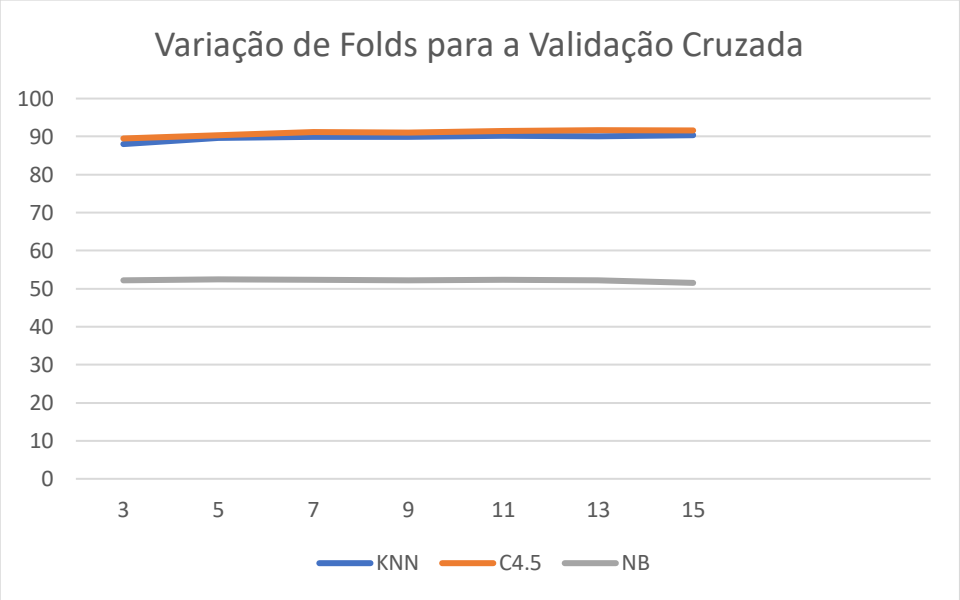

Figura 4.2 - Variação da acurácia em relação ao número de folds.

Na validação cruzada os testes foram feitos com *folds* de 3 a 15. Quanto menor o número de *folds*, uma parcela maior de dados será usada nos testes, logo, poucos dados de treinamento poderão levar a um mal desempenho do algoritmo, porém para a base de dados utilizada a diferença da acurácia conforme o número de *folds* foi baixa porquê o algoritmo foi estruturado com estratégia de embaralhamento. Com o embaralhamento, mesmo com poucos *folds*, as amostras obtidas foram suficientemente representativas para levar a um bom resultado.

5. Considerações Finais

Tendo em vista que a inclusão de novas ferramentas tecnológicas para auxiliar na conversação da comunidade surda e ouvintes, torna-se cada vez mais importante, proporcionando a inclusão no seu contexto social. O desenvolvimento do presente trabalho possibilitou uma análise para a construção de um aplicativo que seja capaz de classificar uma base de dados e, ser utilizada em trabalhos futuros a fim de incluir cada vez mais a comunidade surda.

Para essa análise, foi desenvolvido um aplicativo que proporciona algumas interessantes funcionalidades como as técnicas para teste e treinamento e a execução dos classificadores simultaneamente. Os resultados obtidos, mostram que para a técnica de teste e treinamento *holdout* e a validação cruzada o classificador C4.5 obteve a maior acurácia com apenas 1% de diferença para o classificador *K-Nearest Neighbors* (KNN) e ambos com uma diferença elástica para o *Naive Bayes* (NB) que alcançou uma taxa de 52.68%. No *holdout*, quanto maior a porcentagem de dados usado para teste menor é a acurácia, com uma observação: a acurácia do algoritmo NB não teve uma considerável alteração, diferente dos outros classificadores que caíram quase que 30%. Já na validação cruzada, foi aplicado o teste estatístico para comparar os classificadores e chegar à conclusão de qual seria o melhor para a base de dados testada, e a conclusão foi que o C4.5 e o KNN são considerados iguais, já o NB é muito diferente em comparação aos outros classificadores.

A ferramenta criada, gera a possibilidade de ser usada em trabalhos futuros de visão computacional e aprendizado de máquina simples e/ou complexos. Em trabalhos futuros e junto ao projeto de iniciação científica pretende-se aumentar o número de algoritmos de classificação e incorporar um desses algoritmos ao sistema de reconhecimento de gestos em tempo real.

Referências Bibliográficas

ARAUJO, C. C. M.; LACERDA, C. B. F. Examinando o desenho infantil como recurso terapêutico para o desenvolvimento de linguagem de crianças surdas. ***Revista da Sociedade Brasileira de Fonoaudiologia**, 13*(2), 186-192. de 2008.

BALLARD, D. H.; BROWN, C. B.: **Computer Vision**, 1982

BAQUETA, J. J; BOSCARIOLI, C.: **Uma Discussão Sobre o Papel das Tecnologias no Ensino Aprendizagem de Alunos Surdos**. Cascavel,: Unioeste, 2011.

BARELLI, F.: **Introdução à Visão Computacional**, 2018.

BASILIO, J. A. M; TORRES, G. A; PÉREZ, G. A; MEDINA, L. K. T; MEANA, H. M.P.: Explicit image detection using YCbCr space color model as skin detection. **Applications of Mathematics and Computer Engineering**, p. 123-128, 2011.

BASTOS, Igor L. O.; ANGELO. Michele F. LOULA; Angelo C. **Recognition of Static Gestures Applied to Brazilian Sign Language**. 2015 p. il.

BRASIL (2002). *Lei Federal N. 10436 de 24 de abril de 2002: **Oficializa a Língua Brasileira de Sinais em território nacional*.** Brasília 2002. Disponível em: <http://www.planalto.gov.br/ccivil_03/LEIS/2002/L10436.htm>. Acessado em: 22 de janeiro de 2019.

BRASIL. (2005). Decreto N. 5626/2005: **Regulamenta a Lei n. 10436/2002, que oficializa a Língua Brasileira de Sinais**. Brasília, 2005. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2004-2006/2005/Decreto/D5626.htm>. Acessado em: 22 de janeiro de 2019.

CAMPOS, Raphael. **Árvores de Decisão**. 2017. Disponível em: <https://medium.com/machine-learning-beyond-deep-learning/%C3%A1rvores-de-decis%C3%A3o-3f52f6420b69>. Acessado em: 29 jan. 2019.

CHANG, G.; JAEWAN,P.; CHIMIN, O.; LEE, C.: A Decision Tree based Real-time Hand Gesture Recognition Method using Kinect. **In Journal of Korea Multimedia Society**, Vol. 16, No. 12, December, pp. 1393-1402, 2013.

CHEN, Y.; GAO, W.; MA, J.: Hand Gesture Recognition Based on Decision Tree. **In Proceedings of the International Symposium on Chinese Spoken Language Processing.** International Speech Communication Association, 2000.

COPPIN, B.: **Inteligência artificial**. Rio de Janeiro: LTC, 2010.

DUDA, R. O.; HART, P. E.: **Pattern classification and scene analysis**, volume 3. Wiley New York, 1973.

ESCALANTE, H. J.; MORALES, E. F.; SUCAR, L. E.: A na¨ıve Bayes baseline for early gesture recognition, **Pattern Recognition Letters**, doi: 10.1016/j.patrec.2016.01.013, 2016.

GONZALEZ, R. C.; WOODS, R. E.: **Digital Image Processing**, 1993.

GONZALEZ, R. C.; WOODS, R. E.: **Digital Image Processing**, 2008.

JAVA. Link: https://www.java.com/ . Acessado em Abril de 2019.

JOHN, G. H.; LANGLEY, P.: Estimating continuous distributions in bayesian classifiers. **In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence**, pages 338-345. Morgan Kaufmann Publishers Inc, 1995.

LI, H.; DU, T.; REN, X.: Gesture Recognition Method Based on Deep Learning. **In 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)**, 2018.

LILLESAND, T. M.; KIEFER, R. W.; CHIPMAN, J. W.: **Remote Sensing and Image Interpretation**, 2004.

LOPES, G. K. F.: **O uso das tecnologias no processo de ensino e de aprendizagem do surdo: Libras em Educação a Distância**, 2017.

MITCHELL, T. M.: **Machine Learning**, 1997.

NIMBALKAR, A. K.; KARHE, R. R.; PATIL, C. S.: Face and Hand Gesture Recognition using Principle Component Analysis and KNN Classifier. **International Journal of Computer Applications** (0975 – 8887), vol 99, no 8, 2014.

NIXON, M.; AGUADO**,** A. **Feature Extraction & Image Processing for Computer Vision. [S.I]**: Academic Press. ISBN 9780123965493, 2012.

OPENCV. Link: https://opencv.org/ . Acessado em janeiro de 2019.

PANDAS. Link: https://pandas.pydata.org/ . Acessado em Abril de 2019.

PAVLOVIC, V. I.; SHARMA, R.; HUAN, T. S. Visual Interpretation of Hand Gestures for Human-Computer Interaction. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol 19, no 7, 1997.

PRINCE, S. **Computer Vision: Models Learning and Inference. [S.I]**: Cambridge University Press, 2012.

PSF (Python Software Foundation). Link: https://www.python.org/ . Acessado em janeiro de 2019.

SCIKIT-LEARN. Link: https://scikit-learn.org . Acessado em Abril de 2019.

SEBE, N.; COHEN, I.; GARG, A.; HUANG, T. S.: **Machine Learning in Computer Vision**. Dordrecht: Springer, 2005.

SHAPIRO, l.; STOCKMAN, G. **Computer Vision. Prentice Hall**. ISBN 9780130307965, 2001.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition (4 ed.).** Academic Press. 2008.

VIOLA, P.; JONES, M.: Rapid object detection using a boosted cascade of simple features**. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. CVPR 2001.

WEKA. Link: https://www.cs.waikato.ac.nz/~ml/weka/index.html . Acessado em Abril de 2019.

YEO, H. S.; LEE, B. G.; LIM, H.: **Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware**. ed. Springer, 29 p. 2013.

ZHANG, H.: **The optimality of naive bayes. In FLAIRS Conference, AAAI Press**, 2014.