UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE CIÊNCIAS AGRÁRIAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

SERGIO SILVA RIBEIRO

EXTRAÇÃO DE CARACTERÍSTICAS DE IMAGENS APLICADA À DETECÇÃO DE GRÃOS ARDIDOS DE MILHO

PONTA GROSSA
2015

SERGIO SILVA RIBEIRO

EXTRAÇÃO DE CARACTERÍSTICAS DE IMAGENS APLICADA À DETECÇÃO DE GRÃOS ARDIDOS DE MILHO

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada, curso de Mestrado em Computação Aplicada, da Universidade Estadual de Ponta Grossa, como requisito parcial para obtenção do título de Mestre.
Orientadora: Profª. Dr. Rosane Falate

PONTA GROSSA
2015

Ficha Catalográfica
Elaborada pelo Setor de Tratamento da Informação BICEN/UEPG

Ribeiro, Sergio Silva
R484 Extração de características de imagens aplicada à detecção de grãos ardidos de milho/ Sergio Silva Ribeiro. Ponta Grossa, 2015.
85f.

Dissertação (Mestrado em Computação Aplicada - Área de Concentração: Computação para Tecnologias em Agricultura), Universidade Estadual de Ponta Grossa.
Orientadora: Profª Drª Rosane Falate.

1.Processamento Digital de Imagens. 2.Mineração de Dados. 3.Milho. I.Falate, Rosane. II. Universidade Estadual de Ponta Grossa. Mestrado em Computação Aplicada. III. T.

CDD: 005.3

SERGIO SILVA RIBEIRO

EXTRAÇÃO DE CARACTERÍSTICAS DE IMAGENS APLICADA À DETECÇÃO DE GRÃOS ARDIDOS DE MILHO

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada, Curso de Mestrado em Computação Aplicada, da Universidade Estadual de Ponta Grossa, como requisito parcial para obtenção do título de Mestre.

Ponta Grossa, 27 de fevereiro de 2015

_____

Profª. Dr. Rosane Falate
Doutor em Ciências UTFPR
Universidade Estadual de Ponta Grossa

_____

Profª. Dr. Alaine Margarete Guimarães
Universidade Estadual de Ponta Grossa

_____

Prof. Dr. Mauro Miazaki
Universidade Estadual do Centro-Oeste

Dedico este trabalho a minha amada esposa Marli e minha filha Rafaella que, além da compreensão pelos momentos de ausência, foram e são minha principal motivação.

AGRADECIMENTOS

À minha orientadora, Profª. Dr Rosane Falate, que de forma muito profissional e paciente me ajudou na condução deste trabalho.

À Profª Drª Maria Salete Marcom Gomes Vaz, que ajudou no direcionamento do tema da minha dissertação durante os trabalhos desenvolvidos na disciplina de pesquisa orientada.

À Profª Drª Alaine Margarete Guimarães, que foi fundamental para o entendimento e desenvolvimento dos trabalhos da minha pesquisa relacionados a mineração de dados.

Ao Prof. Dr. Alceu de Sousa Britto Jr., que me ajudou a ampliar minha visão sobre as aplicações de técnicas de PDI, principalmente pela grande contribuição ao meu trabalho, no uso do Padrão LBP.

Ao Prof. Dr. David de Sousa Jaccoud Filho e Drª Luciane Henneberg, do Departamento de Fitotecnia e Fitossanidade, da UEPG, pelas orientações e ajuda.

Aos demais professores do Programa de Mestrado em Computação Aplicada, da Universidade Estadual de Ponta Grossa que, por meio de suas aulas, me ajudaram a aprimorar os conhecimentos em computação.

Aos colegas Marcio Hosoya Name e Teruo Matos Maruyama que, além das ajudas relacionadas a PDI, também foram parceiros na produção de trabalhos publicados.

Aos alunos de iniciação científica do Grupo CAFIRA, que me auxiliaram ao longo do trabalho.

Aos demais colegas do Mestrado, pelos momentos agradáveis e troca de experiências ao longo das aulas.

À UEPG, por disponibilizar toda a sua estrutura, tanto para o programa de mestrado quanto para esta pesquisa, bem como pela qualidade dos professores, comprovada nas aulas.

À CAPES, pela bolsa de estudos durante o último ano do mestrado.

E finalmente, meu agradecimento a todos que contribuíram direta ou indiretamente para a realização deste trabalho.

Cada sonho que você deixa pra trás, é um
pedaço do seu futuro que deixa de existir

(Steve Jobs)

RESUMO

O milho é uma importante cultura em nível nacional e internacional. Seu valor de mercado é influenciado pela qualidade dos grãos. Os grãos ardidos são um problema econômico e sanitário. Rigorosas leis determinam o processo de classificação dos grãos em sadios e ardidos, entretanto este procedimento é feito de forma manual, sujeito à subjetividade e erros humanos. O objetivo deste trabalho foi aplicar métodos computacionais de processamento digital de imagens para extração de características dos grãos de milho. Este trabalho também propôs identificar quais métodos e algoritmos de mineração de dados são mais eficientes para a resolução do problema de identificação dos grãos ardidos de milho. Nesta pesquisa foram usadas amostras de grãos de milho de diferentes cooperativas da região centro oeste do Paraná. Essas amostras foram classificadas por técnicos credenciados nessas cooperativas e submetidas, por meio de programas proprietários e desenvolvidos neste trabalho, aos processos de aquisição de imagens, segmentação e extração de características de cor e textura. Para o desenvolvimento dos programas foi utilizado a linguagem Python em conjunto com o framework SimpleCV. Os dados extraídos das imagens foram armazenados em um arquivo compatível com a ferramenta Weka, que foi utilizado para treinamento e teste da base empregando os métodos de Divisão de Amostras e Validação Cruzada. Todos os algoritmos da ferramenta foram utilizados para processamento dos dados e 24 deles chegaram a uma taxa de acurácia igual e/ou superior a 99%. Os melhores trabalhos correlacionados estudados obtiveram uma taxa de acurácia de 93% para Steenhoek et al. (2001) e 98% para Draganova et al. (2010a). Entre os algoritmos com melhor acurácia foi gerado um modelo, que foi implementado e testado em um programa também desenvolvido neste trabalho, e que chegou a uma acurácia de 99,8%. Os melhores resultados foram obtidos com a combinação do espaço de cor HSV e características de textura com o padrão LBP em imagens com resolução de 300 dpi.

Palavras-chave: Processamento Digital de Imagens, Mineração de Dados, Milho.

ABSTRACT

Maize is an important crop in national and international level. Its market value is influenced by the quality of the grains. The rot (damaged) grains are an economic and health problem. Strict laws determine the process of classification of grains in healthy and rot, however this procedure is done manually, subject to subjectivity and human errors. The objective of this study was to use computational methods of digital image processing for feature extraction of maize grains. This work also proposed to identify which methods and data mining algorithms are more efficient to solve the problem of rot grain maize identification. For this research it was used corn grain samples from various cooperatives in the Midwest Parana area. These samples were classified by certified technicians in these cooperatives and submitted, with manufacturer and developed programs, to the process of image acquisition, segmentation and extraction of colour and texture characteristics. For the development of programs was used Python together with SimpleCV framework. The extracted image data were saved in a file format of the Weka tool that was used to train and to test the base by employing the methods holdout and cross-validation. All tool algorithms were used for data processing and 24 of them have reached a rate equal accuracy and / or greater than 99%. The best studied related work achieved an accuracy rate of 93% for Steenhoek et al. (2001) and 98% for Draganova et al. (2010a). Between the algorithms with the better results, it was chosen one to generate a model that was implemented and tested with program developed in this work. This model obtained a accuracy rate of 99,8% with this model. The best results were obtained with the combination of HSV color space, and texture characteristics with LBP pattern, and images with a resolution of 300 dpi.

Keywords: Digital Image Processing, Data Mining, Maize.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Técnicas de Processamento Digital de Imagens (PDI) têm sido aplicadas em, praticamente, todas as áreas, porque permitem o estudo de fenômenos complexos que não poderiam ser realizados por outros meios convencionais, com a vantagem de serem métodos não invasivos e não destrutivos, em sua maioria, preservando o objeto de estudo. O PDI tem evoluído , com aumento no nível de interesse em morfologia matemática, redes neurais, processamento de imagens coloridas, compressão de imagens e reconhecimento de imagens, em sistemas de análise, baseados em métodos de mineração de dados (WIGGERS et al., 2013).

Assim como ocorre em vários setores da sociedade, os computadores têm se tornado ferramentas indispensáveis para o setor agrícola. Suas aplicações são variadas e envolvem a automação de tarefas manuais, sistemas de gerenciamento remoto, software de gestão, agricultura de precisão, sistema para detecção de doenças, análise climática, entre outras (COCARO & JESUS, 2008; MACHADO, 1998; MEIRA et al., 1996). É importante destacar que muitos desses sistemas fazem uso de imagens e técnicas de PDI para o seu processamento.

A agricultura tem sido impulsionada pelo crescimento populacional, que tem acarretado uma maior demanda pela produção de alimentos. De fato, a produção mundial de grãos é fundamental para atender a crescente demanda de alimentação (SCOLARI, 2006). Além disso, a necessidade de diminuição do impacto ambiental e a diminuição de áreas cultiváveis exigem técnicas que contribuam para o aumento produtivo. Países tecnologicamente desenvolvidos produzem mais que países que sub existem com modelos agrícolas ultrapassados. Essa tecnologia está ligada ao desenvolvimento de insumos, máquinas, estudos de solo e, sobretudo, a produção de sementes e grãos (CARNEIRO, 2005). O uso de sementes selecionadas, além de proporcionar padronização na produção, permite o aumento da qualidade do que é cultivado. Para tanto foi criado no Brasil, pelo Ministério da Agricultura, Pecuária e Abastecimento (MAPA) um manual de análise sanitária de sementes. As sementes são um dos principais recursos para a produção agrícola, e a busca por sementes sadias – livres de pragas - é um dos desafios enfrentados pela agricultura. Desse modo, esse manual busca apresentar as principais doenças que atacam as sementes e os meios de detecção destas (BRASIL, 2009).

O milho esta entre os grãos de maior importância e projeções internacionais indicam aumentos significativos em sua produção a médio e longo prazo (CGIAR, 2014). Os grãos de milho podem ter seus valores econômicos e nutricionais prejudicados por doenças ou manejo inadequado (PINTO, 2005) Devido a isso, existe um processo de classificação e controle antes que os grãos possam ser colocados no mercado. Entretanto, este processo hoje é feito de forma manual e esta sujeito a erros e subjetividade humana. O uso da computação, com técnicas de processamento digital de imagens e algoritmos de aprendizado de máquina podem ser aplicados para auxiliar nos processos de classificação e controle mencionados (POTTER et al., 2012).

Dessa forma, são objetos de interesse deste trabalho o uso do processamento digital de imagens, associado à mineração de dados, como ferramenta de apoio para tomada de decisão da ocorrência ou não de grãos ardidos de milho.

1.1. OBJETIVO GERAL

O objetivo geral desta dissertação é demonstrar o uso de métodos computacionais relacionados ao processamento digital de imagens, em conjunto com técnicas de mineração de dados, para detecção de grãos ardidos em milho.

1.2. OBJETIVOS ESPECÍFICOS

Como objetivos específicos estão:

- Demonstrar diferentes meios de detecção de características em grãos de milho através de PDI;
- Elaborar e apresentar um método para aquisição de imagens para sementes de milho;
- Elaborar e apresentar uma solução para segmentação e extração de características baseada no padrão de cor e textura;
- Elaborar e apresentar solução para conversão dos dados extraídos das imagens para o padrão de mineração de dados;
- Identificar e apresentar os métodos de mineração de dados mais eficientes para a detecção de grãos ardidos de milho; e
- Validar os melhores métodos e analisar os resultados.

1.3. ESTRUTURA DO TRABALHO

Este trabalho está estruturado em 5 (cinco) capítulos, incluindo este capítulo introdutório. No Capítulo 2 é apresentada uma revisão bibliográfica de trabalhos relacionados com a pesquisa desenvolvida. No Capítulo 3 são descritos os materiais utilizados bem como a metodologia aplicada. Os resultados e discussões são apresentados no Capítulo 4. Por fim, no Capítulo 5, são apresentadas as conclusões e perspectivas de trabalhos futuros. Integram este trabalho os Apêndices A, B e C. No Apêndice A é apresentado o código do programa SEGMENTADOR que foi utilizado para o processo de segmentação das imagens. No Apêndice B é apresentado o código do programa EXTRATOR, responsável pela extração de características das imagens. E, por fim, no Apêndice C é apresentado o código do programa TESTMODEL, que implementa e testa um dos modelos de classificação de dados, gerado no processo de mineração de dados.

2. REVISÃO BIBLIOGRÁFICA

2.1. CULTURA DO MILHO

O milho (*Zea mays*) é um alimento importante para a dieta animal (na forma de ração) e humana. O peso médio individual do grão varia de 250mg a 300mg, sendo basicamente composto de amido, proteína, fibra e óleo. É um alimento energético e seus derivados são utilizados na composição de vários produtos na indústria alimentícia, como margarinas, molhos de salada, óleos comestíveis, pães, etc. Também, é utilizado na indústria química e farmacêutica na produção de combustíveis, tintas, inseticidas, cosméticos, cremes de barbear, xaropes, antibióticos, entre outros (CRUZ et al., 2011; PAES, 2006).

O Brasil é um dos maiores produtores de milho no mundo, onde a produção desta cultura foi de 78,5 milhões de toneladas em 2013 e está estimada em 93,6 milhões de toneladas em 2022/23. O consumo interno em 2013 desse alimento foi de 66,7% de sua produção, com uma exportação de 18 milhões de toneladas. Para 2022/2023 é estimado um aumento na exportação para 24,74 milhões de toneladas (BRASIL, 2013). Por volta de 2025 o milho deverá se tornar a cultura com maior produção nos países em desenvolvimento e até 2050, sua produção no mundo deverá dobrar (CGIAR, 2014).

O grão de milho pode ter seu valor nutricional, sanitário e econômico prejudicados por fatores como quebra do grão, rachadura e podridão, entre outros (PINTO, 2005). Além disso, devido à presença de fungos, pode ocorrer o surgimento de micotoxinas que quando ingeridas, podem provocar graves doenças (VIANA, 2009).

2.1.1 Grãos ardidos

Grãos ardidos de milho (Figura 1) são grãos desta cultura que possuem pelo menos um quarto da superfície com descoloração, cujo matiz pode variar do marrom claro ao roxo ou do vermelho claro ao vermelho intenso; a causa da descoloração está relacionada à podridão das espigas acometidas por fungos (PINTO, 2005). Os fungos mais frequentemente encontrados nos grãos ardidos são *Fusarium verticilioides*, *Diplodia maydis* (*Stenocarpella maydis*) e *Diplodia macrospora* (*Stenocarpella macrospora*), atuando de forma deletéria na qualidade dos grãos (MENDES et al., 2012).

Figura 1 – Grãos ardidos de milho.

Fonte: Autor

A partir de setembro de 2013 entraram em vigência as instruções normativas (I.N.) MAPA nº 60/2011 e 18/2012, que estabeleceram o limite máximo de tolerância de 1,0% para os grãos ardidos. Segundo a instrução normativa (I.N.) MAPA nº 60/2011, Artigo 3º, Parágrafo III, Item a, são considerados grãos ardidos “os grãos ou pedaços de grãos que apresentam escurecimento total, por ação do calor, umidade ou fermentação avançada atingindo a totalidade da massa do grão, sendo também considerados como ardidos, devido à semelhança de aspecto, os grãos totalmente queimados”. (CONAB, 2013).

Atualmente a inspeção e classificação de grãos de milho são feitas visualmente por técnicos classificadores, estando sujeitas às seguintes desvantagens: a análise é dependente da interpretação subjetiva do avaliador, de modo que diferentes avaliadores podem apresentar diagnósticos diferentes para uma mesma amostra; o processo de classificação é lento, pois os grãos são avaliados individualmente e a qualidade da análise é influenciada pela fadiga consequente do turno de trabalho que se degrada ao longo do dia (BROSNAN & SUN,2004).

2.2. PROCESSAMENTO DIGITAL DE IMAGENS

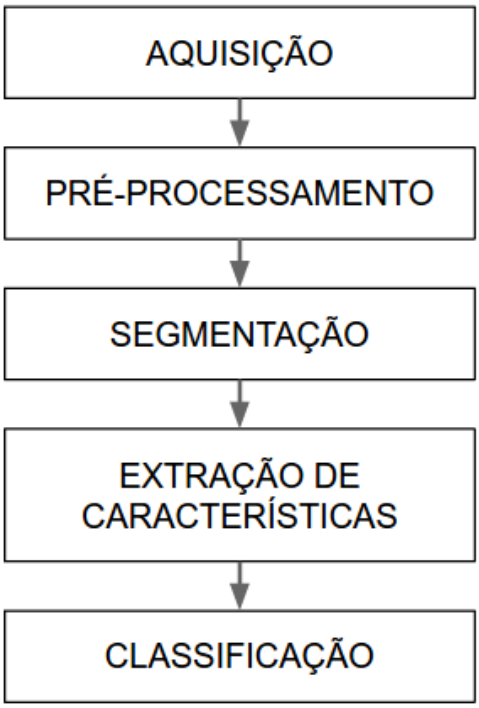
Desde a década de 1920, técnicas e sistemas de PDI têm sido adotadas para, entre outras, o melhoramento da qualidade de reprodução (midiática) das mesmas, bem como a velocidade de processamento das informações. Naquela década, o Sistema Bartlane de transmissão de imagens foi capaz de reduzir o tempo de transporte de uma imagem entre Londres e Nova York de uma semana para três horas. As imagens eram codificadas por meio de um equipamento especializado de

impressão, transmitidas via cabo e decodificadas por um terminal receptor (GONZALEZ e WOODS, 2008). Com a evolução dos computadores, o processamento de imagens digitais passou de horas para segundos. Por exemplo, hoje é possível a transmissão da imagem de maneira quase instantânea, pois, com a utilização de uma câmera digital, pode-se tirar uma fotografia e transmití-la quase instantaneamente a uma ou mais pessoas, em qualquer parte do mundo.

No campo científico, o PDI pode ser entendido como a análise e a manipulação de imagens por computador. PDI envolve, principalmente, a identificação e a extração de informações presentes em uma imagem, para posterior processamento e análise, e isto é importante, já que o sistema de visão humano frequentemente não é capaz de processar o volume de informações que se encontram presentes em uma imagem (ALVARENGA et al., 2005).

Um sistema de PDI pode ser dividido em cinco etapas principais, Figura 2: aquisição, pré-processamento, segmentação, extração de características e classificação, Figura 2 (FALCAO & LEITE, 2006).

Figura 2 – Principais etapas de um sistema de PDI.



Fonte: Autor

A aquisição é a primeira etapa, na qual é feita a aquisição da imagem com algum dispositivo. No pré-processamento são realizadas tarefas de melhoramento da imagem. A etapa de segmentação tem como objetivo a identificação de objetos ou área de interesse. A partir dos objetos identificados na segmentação são

extraídas características para a etapa de análise ou classificação (MARQUES FILHO & VIEIRA NETO, 1999). Dentro de cada uma das etapas de um sistema de PDI são realizadas tarefas específicas como controle de iluminação (aquisição), aplicação de filtros (pré-processamento), identificação de bordas (segmentação), entre outras (JAHNE, 2005).

2.2.1 Cor

A luz é uma onda eletromagnética, ou seja, uma onda composta por campos elétricos e campos magnéticos alternados. Em um determinado intervalo de frequências do espectro eletromagnético, correspondente à luz visível, cada frequência vai resultar em uma sensação de cor (HALLIDAY, 2009). A Colorimetria é a ciência que estuda a cor do ponto de vista físico, e a fotometria é a ciência que a estuda do ponto de vista perceptual (SCURI, 1999).

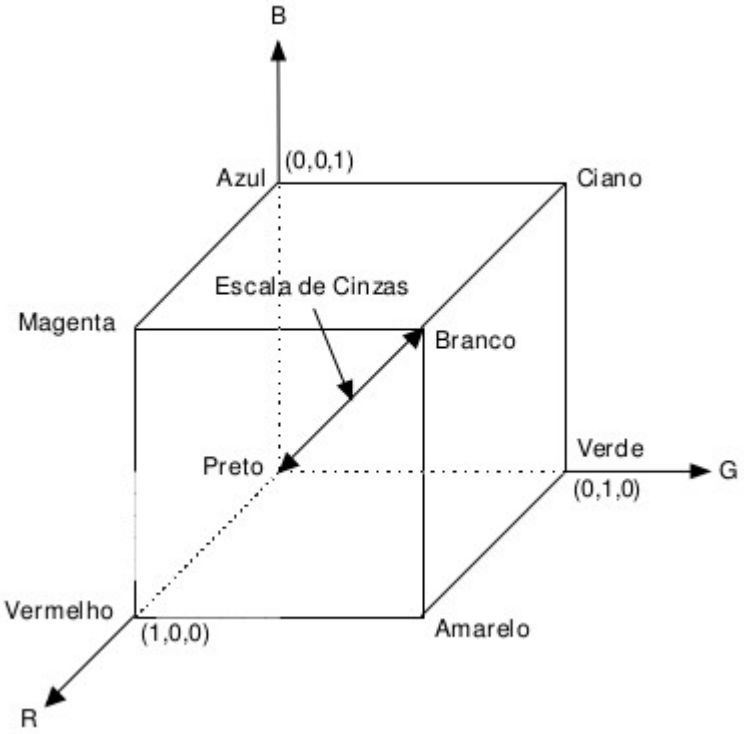
Os modelos ou padrões de cores permitem a especificação das cores em um formato padronizado. Um padrão de cores é uma representação tridimensional, na qual cada cor é representada por um ponto em um sistema de coordenadas. Os padrões mais utilizados são: RGB, CMY, CMYK, YCbCr e HSI ou HSV (MARQUES FILHO & VIEIRA NETO, 1999). Entretanto existem outros padrões como: CIELAB, Lab, XYZ e xyY (Dragavova et al., 2010a). A seguir são descritos os padrões padrões RGB e HSV que foram objetos de estudo deste trabalho.

2.2.1.1 O padrão RGB

RGB é uma sigla para um padrão de cores criado para uso em objetos emissores de luz como monitores, câmeras digitais, televisores, entre outros. A sigla RGB é formada pelas iniciais das cores aditivas *Red* (vermelho), *Green* (verde) e *Blue* (azul) (SCURI, 1999).

O padrão RGB é baseado em um sistema de coordenadas cartesianas, como um cubo, onde três de seus vértices correspondem às cores primárias R, G e B (Figura 3). As outras cores são formadas pela combinação dos valores R, G e B, sendo que no vértice junto à base do cubo (na origem do sistema de coordenadas) está o preto, correspondendo à ausência de valores de cor, e o vértice na outra extremidade da diagonal principal, contendo a origem, está o branco (MARQUES FILHO & VIEIRA NETO, 1999).

Figura 3 – Padrão RGB.



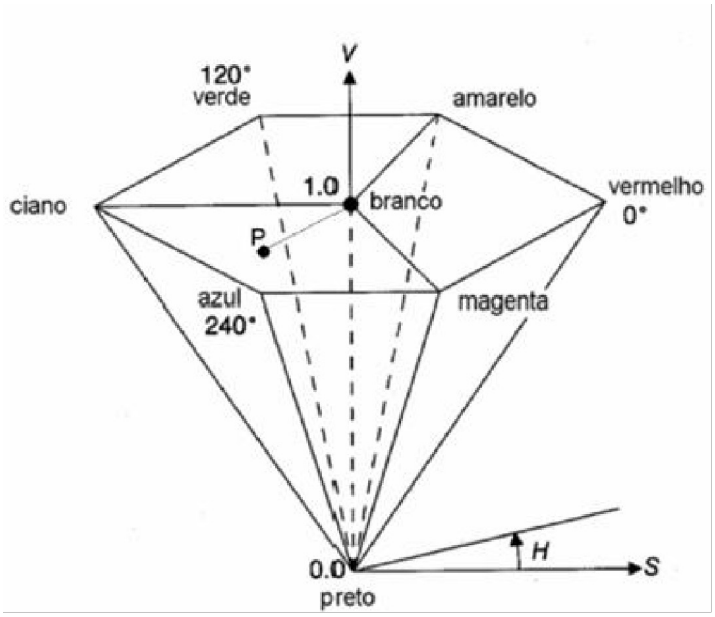
Fonte: Marques Filho & Vieira Neto (1999).

O valor de cada uma das cores primárias, no sistema decimal, varia de 0 a 1, correspondendo a sua tonalidade. Entretanto, a forma mais comum de representação desse valor é o uso dos inteiros de 0 a 255, devido ao fato de comumente cada valor de um canal de cor ser armazenado em um byte (8 bits) (GONZALEZ e WOODS, 2008).

2.2.1.2 O padrão HSV

O padrão HSV, é um padrão de cores formado pelas componentes matiz (*hue*), saturação (*saturation*) e valor do brilho ou intensidade luminosa (*value* ou *intensity*), Figura 4.

Figura 4 – Representação gráfica do padrão de cores HSV.



Fonte: Adaptado de Shapiro & Stockman, 2001.

A matiz ou tonalidade define qual a cor do espectro. Os valores da matiz são entre 0 e 360, ou entre 0 e 100% em algumas aplicações. Com relação à saturação ou pureza, a faixa de valores varia de 0 a 100% e, quanto menor for este valor, mais para o tom de cinza será a cor e, quanto maior for este valor, mais “pura” é a cor do espectro. O último componente, o valor do brilho ou a intensidade luminosa, define o brilho da cor que pode assumir valores entre 0 e 100% (SHAPIRO & STOCKMAN, 2001).

O padrão de cor HSV é uma transformação não linear do padrão RGB, dada pela Equação 1 (OHTA et al., 2006; KANADE et al., 2005).

$$H = \left(60\frac{(G-B)}{(MAX-MIN)}\right) + 0, se\ MAX = R\ e\ G \geq B \quad (1)$$

$$H = \left(60\frac{(G-B)}{(MAX-MIN)}\right) + 360, se\ MAX = R\ e\ G < B$$

$$H = \left(60\frac{(G-B)}{(MAX-MIN)}\right) + 120, se\ MAX = G$$

$$H = \left(60\frac{(G-B)}{(MAX-MIN)}\right) + 240, se\ MAX = B$$

$$S = \frac{(MAX-MIN)}{MAX}$$

$$V = MAX$$

Na Equação 1 os valores de *R*, *G*, e *B* são os valores individuais (entre 0 e 1) para cada um dos canais R, G e B, do padrão RGB e, *MAX* e *MIN* são os valores máximos e mínimos de cada canal (OHTA et al., 2006).

2.2.2 Imagem

Uma imagem digital é o objeto principal de um sistema de PDI. Formada por vários pixels, sua menor unidade, pode ser definida matematicamente como uma função de *f(x,y)*, onde *x* e *y* correspondem as coordenadas de cada pixel que compõe a imagem. Em uma imagem monocromática, cada pixel recebe um valor que corresponde à intensidade luminosa naquele ponto, referente a uma escala de tons de cinza ou brilho. No caso de imagens coloridas, cada pixel é formado por um conjunto de valores. É o que ocorre com imagens no padrão de cores RGB (Seção 2.2.1.1) que contém uma função *f(x,y)*, ou valor, para cada banda ou canal, R (*red* – vermelho), G (*green* – verde) e B (*blue* – azul), respectivamente (MARQUES FILHO & VIEIRA NETO, 1999).

Ao ser digitalizada uma imagem, seu tamanho passa a ser adimensional, ou seja, não possui uma unidade física que o defina. Entretanto, pode-se estimar a medida dos elementos da imagem determinando o tamanho de cada pixel, pela razão entre o número de pixels e o tamanho da imagem real, no filme ou dispositivo. Essa razão chama-se resolução e, em geral, é calculada em pontos por polegada (*dots per inch* - DPI), ou em alguns casos pontos por centímetro (*dots per centimeters* - DPC) (SCURI, 1999).

2.2.3 Aquisição

A aquisição da imagem é a primeira etapa de um sistema de PDI (Figura 2, seção 2.2), onde uma cena é captada do mundo real e convertida para o formato digital; todas as etapas posteriores dependem desta etapa inicial (BURGER & BURGE, 2007). Um sistema de aquisição é composto, basicamente, de dois elementos: um sensor, que faz a captação do sinal eletromagnético e produz um sinal elétrico equivalente; e um digitalizador, responsável em fazer a conversão do sinal elétrico para a forma digital (GONZALEZ e WOODS, 2008). Câmeras e scanners são exemplos de sistemas de aquisição.

2.2.4 Pré-processamento

A imagem digital gerada pela etapa de aquisição pode apresentar diversas imperfeições, como pixels com valores distorcidos; problemas de brilho ou contraste; descontinuidade de informações, como caracteres interrompidos ou indevidamente conectados. Esses problemas ou ruídos podem ser provenientes de diversas fontes, como o tipo de sensor utilizado, a iluminação do ambiente, as condições climáticas no momento da aquisição da imagem, a posição relativa entre o objeto de interesse e a câmera. O ruído não é apenas uma interferência no sinal de captura da imagem, pode ser também interferências que prejudicam a interpretação ou o reconhecimento de objetos na imagem (MARENGONI & STRINGHINI, 2009).

A etapa de pré-processamento tem como objetivo principal a utilização de técnicas computacionais aplicadas às imagens para o melhoramento ou a correção de imperfeições. Entre ferramentas básicas para remoção de ruídos estão os filtros. Filtros espaciais atuam diretamente na imagem, ou seja, as técnicas de filtragem no domínio espacial são aquelas que atuam diretamente sobre a matriz de pixels, que é a imagem digitalizada. As funções de processamento de imagens no domínio espacial podem ser expressas como na Equação 2 (MARQUES FILHO & VIEIRA NETO, 1999).

$$g(x,y)=T[f(x,y)] \tag{2}$$

onde $g(x,y)$ é a imagem processada, $f(x,y)$ é a imagem original, e T é um operador em f, definido em uma certa vizinhança de (x,y).

Na Figura 5 é apresentado um exemplo de imagem com ruído (Figura 5(a)) e de uma imagem que recebeu o processo de filtragem (Figura 5(b)). Verifica-se, nesse caso, que as linhas diagonais existentes na Figura 5(a) foram suavizadas na Figura 5(b) (MARENGONI & STRINGHINI, 2009).

Figura 5 – Imagem (a) com ruído e (b) após filtragem.

(a) (b)

Fonte: Marengoni & Stringhini, 2009.

Entre os filtros existentes, tem-se o filtro da mediana, no qual os valores dos pixels contidos em uma determinada região (máscara) são ordenados e o valor que ocupa a posição mediana é selecionado para a posição *(x, y)* da imagem filtrada. Esse método não linear apresenta bons resultados, pois reduz o efeito de ruído de pulso (Figura 6), do tipo “sal e pimenta” (*salt and peper)*, uma vez que valores pontuais raramente aparecem juntos e, portanto, nunca ocupam a posição mediana (MARENGONI & STRINGHINI, 2009).

Figura 6 – (a) Imagem com ruído impulsivo (sal e pimenta), e (b) a mesma imagem após aplicação do filtro da mediana.

(a) (b)

Fonte: Marques Filho & Vieira Neto, 1999.

Segundo Marques Filho & Vieira Neto (1999), a mediana *m* de um conjunto de *n* elementos é o valor tal que metade dos *n* elementos do conjunto ficam abaixo de *m* e a outra metade acima de *m*. Se o valor de *n* é ímpar, a mediana é o próprio elemento central do conjunto ordenado e quando *n* é par, a mediana é calculada pela média aritmética dos dois elementos mais próximos do centro.

Outro filtro utilizado em PDI é o filtro da média. Para o caso de uma máscara, ou um número de pixels envolvidos igual a 8, o pixel central, inicialmente de valor 11, Figura 7(a), receberá o valor médio dos valores dos 8 pixels vizinhos a ele, Figura 7(b), ou seja, o valor 7, Figura 7(c).

Figura 7 – Calculo filtro da média; (a) Imagem original; (b) calculo da média com 8 pixels vizinhos e (c) imagem resultante.

1	10	1
14	11	8
10	1	7

(a)

(1+10+1+14+8+10+1+7)/8=7

(b)

	7	

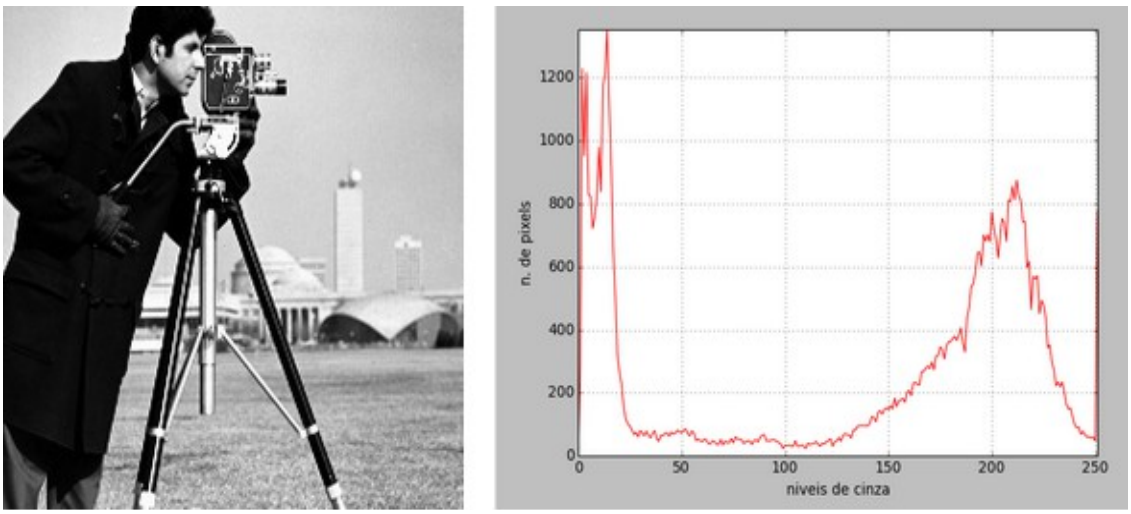
(c)

Fonte: Autor.

2.2.5 Histograma

O histograma de uma imagem é o conjunto de números que indica o percentual de pixels presentes naquela imagem em relação a um determinado nível de cinza ou tom de um determinado canal de cor (Figura 8). Esses valores, em geral, são apresentados em um gráfico de barras que fornece, para cada nível de cinza ou tom de um determinado canal de cor, o número ou percentual de pixels correspondentes àquele nível de cinza ou tom. Dessa forma, a partir do histograma é possível verificar uma concentração maior ou menor para alguns níveis de cinza (ou tom) e sua distribuição (DEMAAGD, 2012).

Figura 8 – (a) Imagem e (b) histograma da imagem.



(a) (b)

Fonte: Adaptado de Gonzalez & Woods (2008).

Imagens escuras tendem a concentrar maiores quantidades ou percentuais de pixels, próximos ao 0 (zero), enquanto imagens claras têm maior número de pixels próximo ao 255 (MARQUES FILHO & VIEIRA NETO, 1999).

O cálculo do histograma é dado pela Equação 3:

$$p(r_k)=\frac{n_k}{n} \tag{3}$$

onde, $p(r_k)$ é a probabilidade de ocorrência de r_k , que corresponde aos níveis de cinza, n_k é o número de pixels cujo nível de cinza (ou tom de um determinado canal) corresponde a k , que é o *k*-ésimo nível de cinza, e n representa o número total de pixels na imagem.

2.2.6 Segmentação

A segmentação é um processo de divisão de uma imagem em regiões significativas. Essas regiões podem ser frontais, de fundo ou um objeto específico a ser detectado na imagem e são definidas com o auxílio de algumas características, como cor, borda ou similaridade (SOLEM, 2012).

Quando o processo de segmentação é baseado em cor, sua eficiência está relacionada ao fato da cor do objeto de interesse ser substancialmente diferente da cor de fundo (DEMAAGD et al., 2012). Nesse caso é possível usar um limiar, um valor ou faixa de valores, para determinar que os pixels acima, abaixo ou entre os valores correspondem à região frontal, de fundo ou o(s) objeto(s) na imagem. Também é possível utilizar um cálculo de distância para fazer a segmentação, como a euclidiana (Equação 4), correspondendo a distância medida através da raiz quadrada da soma dos quadrados das diferenças entre as coordenadas dos pontos em questão (MARQUES FILHO & VIEIRA NETO, 1999).

$$D_e(A, B)=\sqrt{(x_a-x_b)^2+(y_a-y_b)^2} \tag{4}$$

onde, $D_e(A, B)$ é o cálculo da distância euclidiana dos pontos A e B, x_a e y_a são as coordenadas do pixel A e x_b e y_b são as coordenadas do pixel B.

A Figura 9(a) apresenta uma imagem contendo folhas e flores, da qual se quer isolar (segmentar) as flores do restante da imagem. Nota-se que as flores têm tons próximos ao branco, logo, um limiar que mantenha na imagem final somente pixels com estes tons resolveria o problema. Entretanto, antes de aplicar esse limiar, foi aplicado um filtro da média na imagem, Figura 9(b). Isso foi feito porque na Figura 9(a) tem algumas regiões da haste e de folhas com tons próximos ao branco também, o que apareceriam na imagem final.

Figura 9 – Exemplo de um processo de segmentação: (a) uma imagem original; (b) imagem com aplicação de filtro; e (c) imagem segmentada.

(a) (b) (c)

Fonte: Solem, 2012.

Com o uso do filtro da média, essas regiões foram eliminadas e, então, o limiar colocado pode ser aplicado com sucesso, Figura 9(c).

2.2.7 Extração de características

Segundo De Queiroz & Gomes (2001), o processo de extração de características é posterior à etapa de segmentação e fornece informações para etapas seguintes, como classificação e análise. Com base nos objetos detectados pelo processo de segmentação, diversas informações ou características podem ser obtidas a partir destes objetos, como histograma, tamanho, área, perímetro, textura, contornos, etc.

Figura 10 – Extração de característica (área) de um morango.

Fonte: Autor.

Na Figura 10 é mostrado um exemplo de extração de características. Nesse caso, foi feita a detecção da região de vermelho presente na imagem que corresponde à maturação do morango. Posteriormente foi feita a quantificação de pixels vermelhos presentes (53203 pixels) que corresponde ao cálculo da área madura do fruto (RIBEIRO et al., 2014).

2.2.8 Classificação

Para o processo de classificação de objetos é necessário identificar padrões a partir de um conjunto de medidas. Cada objeto é um padrão e os valores medidos são as características deste padrão. Sendo assim, um conjunto de objetos similares, com uma ou mais características em comum, é considerado como pertencente à mesma classe de padrões. Há diversos tipos de características que podem ser usadas para buscar determinadas particularidades de um objeto na imagem (contorno, área, volume, altura, largura, textura, etc.) e cada uma destas características é obtida por meio de técnicas computacionais específicas (DE QUEIROZ & GOMES, 2001).

2.3. PADRÃO BINÁRIO LOCAL

Para a extração da textura de objetos de interesse da imagem pode ser utilizado um operador Padrão Binário Local (*Local Binary Pattern* - LBP), que calcula, para cada pixel, uma conFiguração sobre sua vizinhança (PIETIKAINEN, 2010). A forma de cálculo comum é utilizar uma vizinhança de oito pixels (Figura 11).

Figura 11 – Vizinhança de oito pixels.

0	1	2
3	0	4
5	6	7

Fonte: Autor.

A ideia é utilizar o pixel central como limiar e comparar com os pixels vizinhos, verificando se cada pixel dos oito vizinhos é maior ou igual ao pixel central. O resultado são oito comparações de verdadeiro ou falso, que são identificadas como zero e um. Cria-se então um código, a partir dessas comparações e, para cada uma delas, é dado um peso, de modo que este código de oito comparações gere um

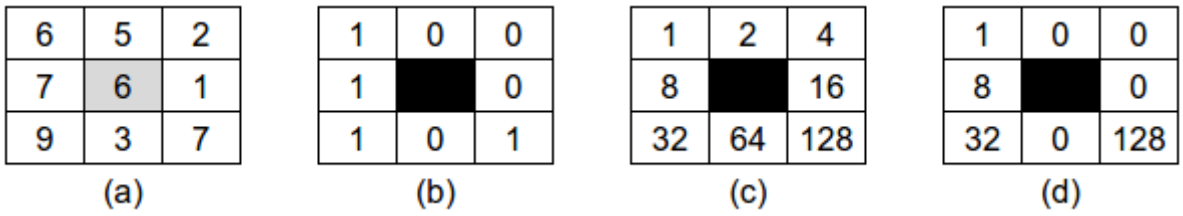
valor entre 0 e 255 (BRAHNAM et al., 2014).

A Equação 5 apresenta a rotina de cálculo proposta por Ojala et al. (1996):

$$LBP = \sum_{i=1}^{7} (f_p \geq f_c) 2^i \quad (5)$$

na qual, $f_p \geq f_c = 1, se\ f_p \geq f_c\ ou\ 0, se\ contrário$,sendo que f_c é o pixel central e f_p é o pixel da vizinhança. Na Figura 12 é apresentado um exemplo da aplicação da Equação 5.

Figura 12 – Exemplo do operador LBP.



Fonte: Ojala et al. (1996).

Na Figura 12(a) é mostrada uma distribuição de pixels 9 x 9, na qual o pixel central possui valor 6. Na Figura 12(b) é mostrado o resultado da comparação do pixel central na Figura 12(a) com seus oito vizinhos; sendo o valor do pixel vizinho maior ou igual ao pixel central, então o resultado é 1 (verdadeiro), caso contrário, o resultado é 0 (falso). Na Figura 12(c) é mostrado o peso atribuído para cada vizinho e na Figura 12(d) é apresentada a multiplicação dos pesos pelo resultado obtido da comparação. O código final gerado (1+0+0+8+0+32+0+128) é utilizado então para calcular valor final de LBP, que é o resultado da soma desses valores, ou seja 169 .

2.4. MINERAÇÃO DE DADOS

A mineração de dados (MD) é o processo de busca de padrões consistentes, que se repetem por meio de regras ou sequências temporais, em um grande volume de dados, utilizando técnicas de estatística, inteligência artificial, recuperação de informação, algoritmos de aprendizagem, redes neurais, rede bayesiana, entre outras (HAN & KAMBER, 2006; DZIEKANIAK, 2010).

A crescente produção de informações, que ultrapassa a capacidade da análise humana, faz com que a MD tenha considerável importância para as organizações (MAIMON & ROKACH, 2010). A descoberta de conhecimento em MD é um processo iterativo, pois determinadas etapas podem necessitar de mais tempo do que o inicialmente esperado e requer a participação de seu utilizador para a

tomada de decisão (FAYYAD et al., 1996).

A mineração de dados utiliza o aprendizado de máquina, onde seus métodos de previsão são definidos como de aprendizagem supervisionada e não supervisionada. Na aprendizagem supervisionada são utilizadas técnicas que tentam descobrir a relação entre os atributos de entrada e saída. Nesse caso, essa relação é representada por modelos, que explicam fenômenos ocultos no conjunto de dados (COSTA, 2011). Esses modelos podem ser do tipo classificação, criando associação entre atributos de entrada e as classes predefinidas, ou de regressão, mapeando um valor de entrada e um valor real (ROCHA et al., 2007). Na aprendizagem não supervisionada são utilizadas técnicas de agrupamento de instâncias, de acordo com medidas de similaridade ou de distância (MAIMON & ROKACH, 2010).

2.4.1 Métodos para treinamento e teste

No contexto de aprendizado de máquina para a classificação é necessário que um modelo tenha sido gerado. Esse modelo é induzido a partir de uma base de treinamento com exemplos, ou seja, valor(es) ou dado(s) de entrada e valor(es) de saída para aqueles dado(s) de entrada. A partir do modelo gerado, novos exemplos de teste podem ser classificados. Existem diversos métodos de treinamento e testes que são utilizados na mineração de dados, entre eles destacam-se o *Holdout* (Divisão de Amostras) e o *Cross-validation* (Validação Cruzada) (QUILICI-GONZALEZ & DE ASSIS ZAMPIROLLI, 2014).

No método Divisão de Amostras os dados são divididos percentualmente em dois subconjuntos, sendo um deles para treinamento (para a construção do modelo) e o outro para teste (PIMENTA et al., 2009). Em geral, os dados são divididos em ⅔ para treinamento, ou seja, 66% da amostra e ⅓ (34%) para teste (QUILICI-GONZALEZ & DE ASSIS ZAMPIROLLI, 2014).

No método de Validação Cruzada, os dados são divididos em subconjuntos. Um dos subconjuntos é utilizado para treinamento, enquanto os demais são utilizados para teste. A Validação Cruzada de 10 *folds* (partições) significa que uma parte é utilizada para treinamento e as nove partes restantes são usadas para teste. Esse procedimento é repetido de forma circular com todas as partes e o resultado final é a média das avaliações (TAVARES et al., 2007).

2.4.2 Ferramenta Weka

A ferramenta Waikato Ambiente para Análise (*Waikato Environment for Analysis* - Weka), construída na Linguagem Java, (HALL et al., 2009), é um software que oferece diversos algoritmos de mineração de dados em seu ambiente, além de apresentar possibilidade de criação de novos algoritmos. Possuindo uma interface amigável e de rápido processamento, os principais algoritmos disponíveis no Weka são: regressão, classificação, mineração de regras de associação, e seleção de atributos (SATO et al., 2013).

A partir dos algoritmos de aprendizagem de máquina, exercitados no conjunto de dados, e sua aplicação, obtêm-se as métricas de classificação correta (*correctly classified*), erro relativo absoluto (*relative absolute error*), tempo de processamento (*time taken*) e o índice Kappa (COSTA et al., 2010; CORSO & GIBELLINI, 2011).

O erro relativo absoluto (*relative absolute error* – RAE) é uma medida utilizada para avaliar a qualidade de um classificador, onde valores menores indicam maior precisão do modelo e o valor próximo de zero indica um modelo estatisticamente perfeito. O RAE é obtido pela Equação 6:

$$RAE = \frac{\sum_{i=1}^{a} |Vp_i - Ve_i|}{\sum_{i=1}^{a} |Ve_i - \bar{a}|} \quad (6)$$

onde, a representa o número de amostras, Vp_i é o valor predito para *i-ésima* amostra, Ve_i é o valor esperado da *i*-ésima amostra e $\bar{a}$ corresponde a média dos valores da amostra (SOUZA, 2011).

O *time taken* é o custo ou tempo de processamento, referente ao tempo em segundos que o algoritmo leva para fazer o treinamento e o teste. Por fim, o índice Kappa, k , permite medir a precisão de uma classe ou o grau de acerto ou concordância desta classe. Os valores de k vão de 0 a 1 e, quanto mais próximo de 1, maior é o grau de acerto da classe. O índice Kappa é obtido pela Equação 7:

$$k = \frac{Pr_a - Pr_e}{1 - Pr_e} \quad (7)$$

onde Pr_a é a concordância observada e Pr_e é a concordância esperada. A Tabela 1 apresenta uma sugestão de interpretação do índice Kappa, proposta por Landis e Koch (1977).

TABELA 1 – Interpretação dos valores do índice Kappa (LANDIS & KOCH, 1977).

k	**Interpretação**
< 0	Nenhuma concordância
0 a 0,2	Leve concordância
0,21 a 0,4	Concordância regular
0,41 a 0,6	Concordância moderada
0,61 a 0,8	Concordância substancial
0,81 a 1	Concordância quase perfeita

Como mencionado anteriormente, valores de k próximos a 1 (0,81 < k < 1) indicam que a classe obtida apresenta uma concordância quase perfeita, enquanto que valores de k próximos a zero (0 < k < 0,2) significa que a classe oferece baixa concordância entre o valor da entrada e o esperado; e índices Kappa menores que zero revelam que a classe em análise não fornece relação entre os dados de treinamento.

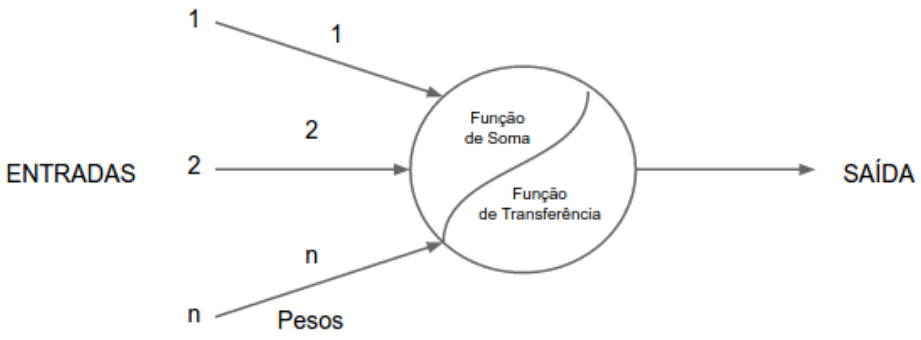
2.4.3 Algoritmos da ferramenta Weka

Neste trabalho foram utilizados todos os algoritmos disponíveis na ferramenta Weka que estão distribuídos nos seguintes grupos: redes neurais, árvore de decisão, máquinas de vetores de suporte, floresta aleatória, bayesianos e regressão.

Os algoritmos de redes neurais são modelos computacionais que identificam as relações em um conjunto de variáveis e/ou descobrem padrões no conjunto de dados baseado no funcionamento biológico dos neurônios, que são definidos como unidades de processamento. Esses neurônios (Figura 13) estão conectados por meio de ligações de entrada e saída, onde cada uma das ligações de entrada possui pesos associados (COSTA & SIMÕES, 2008).

Na Figura 13 é apresentado um exemplo de um neurônio artificial com *n* entradas. A saída do neurônio depende da soma, ou do resultado da função de transferência, obtida dos produtos entre o valor da entrada e o peso associado.

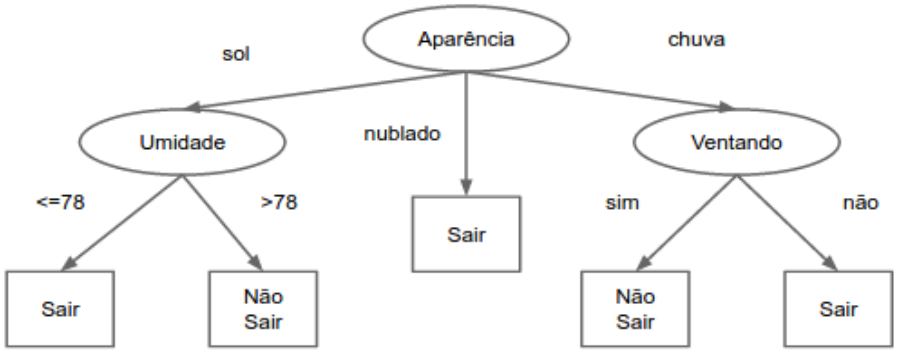
Figura 13 – Exemplo de um neurônio artificial.



Fonte: Autor.

Um algoritmo de árvore de decisão é formado por nós, partindo de um nó raiz e, com exceção deste, cada um dos nós intermediários especifica um teste para o atributo com seu valor especificado pelo ramo. Esse conjunto de regras é seguido até atingir o nó terminal ou folha. Em resumo, uma árvore de decisão possibilita dividir, recursivamente, um conjunto de dados de treino, até que seja fornecida uma classificação para a instância (TURBAN et al., 2010). A Figura 14 apresenta um exemplo de árvore de decisão usada para determinar sobre a saída ou não de um determinado local com base na observação da aparência do tempo.

Figura 14 – Exemplo de árvore de decisão para sair ou não de um determinado local.



Fonte: Autor

Como pode ser visto na Figura 14, com base na informação do nó raiz “Aparência”, que pode ser “sol”, “chuva”, e “nublado”, é possível ir para nós intermediários, “umidade” e “ventando”. No nó “umidade” é realizado um teste de decisão com base no valor da umidade, que resulta em “Sair”, caso os valores de umidade sejam menor ou igual a 78, e de “Não Sair”, para os demais valores. No nó intermediário “ventando” o teste é lógico, ou seja, se estiver ventando (“sim”), a decisão é “Não Sair” e, caso contrário, retorna-se “Sair”. No terceiro caso da

Aparência, o “nublado”, tem-se sempre o resultado de “Sair”.

Máquina de vetores de suporte (*Support Vector Machines* - SVMs) é uma técnica baseada na teoria de aprendizado estatístico e reúne um conjunto de métodos do aprendizado supervisionado para análise de dados e reconhecimento de padrões (LORENA & DE CARVALHO, 2007). As SVMs usam um mapeamento não linear para transformar os dados de treino inicial em uma dimensão superior e, nesta nova dimensão, é procurado por um hiperplano de separação ótimo linear (HAN & KAMBER, 2006). Como em outros algoritmos de mineração de dados, a SVM constrói um modelo baseado em um conjunto de dados utilizado para treinamento. Na etapa de testes, a SVM padrão recebe como entrada um conjunto de dados e prediz para qual classe a instância pertence. A eficiência das máquinas de vetores de suporte é comparável aos das redes neurais artificiais, apresentando, em algumas tarefas, resultado superior (PONTIL & VERRI, 1998).

A floresta aleatória (*Random Forest* - RF) é uma combinação de diferentes árvores de decisão, onde vetores aleatórios são amostrados e distribuídos igualmente e de forma aleatória para todas as árvores da floresta. Após a geração de certa quantidade de árvores, cada uma delas, tendo como referência o valor de entrada, lança um voto para uma classe do problema. Posteriormente, a classe mais votada é a escolhida para predição do classificador. Os algoritmos de floresta aleatória apresentam um bom funcionamento em comparação com outros classificadores e são considerados de fácil implementação, pois possuem apenas dois parâmetros: a quantidade de variáveis no subconjunto aleatório de cada nó e a quantidade de árvores na floresta (BREIMAN, 2001).

O algoritmo de *Naive bayes* é um classificador probabilístico simples baseado no teorema de Bayes (JOHN & LANGLEY, 1995). Esse método assume que os atributos em uma determinada classe são independentes dos valores de outros atributos, para formar estimativas probabilísticas para classificação (WITTEN & FRANK, 2005; HAN & KAMBER, 2006).

A regressão múltipla (*Multiple Regression*) é uma metodologia estatística de previsão, responsável por modelar uma relação linear entre um conjunto de variáveis independentes (entradas) e uma variável dependente (saída). Sua aplicação permite estimar o valor de uma variável, com base em um conjunto de outras variáveis, e, quanto mais significativo o peso de uma variável ou de um conjunto de variáveis, melhor a resposta, devido à natureza linear do método (KASZNAR & GOLÇAVES,

2011).

A regressão logística (*Logistic Regression*) é um método estatístico responsável por produzir um modelo de predição de valores recebidos por uma variável categórica, em geral, binária (DAYTON, 1992). Diferente das variáveis contínuas, as variáveis categóricas assumem alguns valores de resposta que podem ser binários (dicotômicos), com apenas dois valores de resposta, ou politômicos, mais de três valores (grupos) de resposta (MENDES & VEGA, 2011). Os algoritmos do grupo de regressão logística são usando para classificação em aprendizado supervisionado.

2.5. TRABALHOS CORRELATOS

Trabalhos relacionados ao uso do processamento digital de imagens para auxiliar na indústria de alimentos e agricultura são relatados desde 1990. Esses trabalhos têm sido desenvolvidos para auxiliar no processo de detecção de doenças, deformidade, classificação, medição, entre outros (BROSNAN & SUN, 2004).

Estudos relacionados à detecção e medição de brancura do grão de milho, danos mecânicos e mofo com auxílio de processamento digital de imagens foram relatados em Liu & Paulsen (1997) e em Ng et al. (1997).

Ni et al. (1997) projetaram um protótipo de automatização do processo de classificação de grãos de milhos com auxílio de sensores óticos e luz estroboscópica (usada para auxiliar na detecção de objetos em movimento). Com o protótipo foi obtida uma taxa de acerto de 91% para detecção de grãos inteiros e 94% para a detecção de grãos quebrados.

Um estudo relacionado à classificação de grãos de milho tendo como referência parâmetros fornecidos pelo Serviço Nacional de Inspeção de Grãos (*Federal Grain Inspection Service* - FGIS), órgão responsável pelas normas de classificação de grãos nos Estados Unidos, semelhante ao MAPA no Brasil, foi apresentado por Steenhoek et al. (2001). Nesse trabalho foi desenvolvido um sistema de aquisição de imagens, utilizando câmera baseada no padrão RGB, e foram feitas aquisições individuais de 720 grãos. Após o processo de segmentação, e a extração das informações dos canais de cores RGB, os autores utilizaram uma rede neural probabilística (*Probabilistic Neural Network* - PNN) para o processo de classificação. O software de classificação utilizado, com a PNN implementada, foi o NeuroShell EasyClassifier (WARD SYSTEMS GROUP, 1997). O sistema

desenvolvido conseguiu uma taxa de acerto de 93% para detecção de grãos danificados.

Um estudo sobre classificação de grãos baseado em reconhecimento de padrões com uso da análise de componentes principais (*Principal Component Analysis* - PCA) foi apresentado por Potter et al. (2012). Nesse trabalho foram utilizadas 20 imagens de grãos para treinamento, adquiridas com sistema próprio, e 400 imagens (200 de grãos sadios e 200 de grãos ardidos) para teste, provenientes de uma base de dados já existente. Os testes foram feitos com os espaços de cores RGB, HSV e CIELAB[1]. Desses três espaços de cor, o RGB e o CIELAB não apresentaram variação significativa e o melhor resultado foi com o espaço de cor HSV, com uma acurácia de 89%.

Draganova et al. (2010a) desenvolveram um sistema automatizado para detecção de *fusarium* em grãos de milho. Os padrões de classificação utilizados no trabalho foram aqueles regulamentados pelo *Bulgarian Standards* (Normas Búlgaras) (BDS 529-84, 1985), órgão equivalente ao MAPA no Brasil; e para a geração do algoritmo de classificação foram usados o PNN e a Lógica Fuzzy[2]. As informações extraídas das imagens para a classificação foram os dados dos espaços de cor HSV, Lab, XYZ, xyY e YCbCr, combinados com características extraídas de textura. Para o experimento foram usados os grãos tipo XM 87, 26A e o híbrido Kneja 613, sendo 100 amostras de cada. O software foi desenvolvido no MATLAB 7.1 (COLEMAN & VAN LOAN, 1988). O melhor resultado foi com a combinação de textura e cor, com o algoritmo PNN como classificador, com uma acurácia de 98% na detecção de *fusarium*.

Draganova et al. (2010b) realizaram um trabalho adicional com abordagem baseada em análise espectral nas regiões do visível e do infravermelho próximo. Nesse trabalho eles usaram a variação espectral, proveniente de espectros de 300 grãos de milho, sendo 150 grãos sadios e 150 de grãos doentes. Para coleta dos

1 CIELAB é um espaço de cor que foi descrito em 1976 pela Comissão Internacional L'Eclairage (CIE), onde L representa a luminosidade, A é a medida do croma no eixo vermelho-verde e B a medida do croma no eixo amarelo-azul (TAKATSUI, 2011).

2 Lógica Fuzzy também conhecida como lógica difusa ou nebulosa, difere da lógica clássica, que permite apenas valores 0 e 1, no sentido de trabalhar com a incerteza na qual são possíveis valores intermediários entre 0 e 1 (WEBER & KLEIN, 2003; CHENCI et al., 2011).

espectros, eles utilizaram o espectrômetro NIRSystem 6500. Para cada grão, foram feitas três leituras do espectro, para a mesma faixa de comprimento de onda, resultando, ao final, 900 espectros. Para a análise espectral utilizaram o software Pirouette Version 3.11 e o software SIMCA[3] para análise dos dados utilizando o PCA[4]. Com essa nova abordagem foi conseguida uma acurácia de 99,3%.

3 Modelagem Independente Suave de Analogia de Classes (*Soft independent modeling of class analogy* – SIMCA) trata-se de uma técnica de reconhecimento de padrões e classificação (WATERBEEMD, 1995).

4 Análise de Componentes Principais (*Principal Component Analisys* – PCA) é um método que faz análise dos dados objetivando sua redução, eliminando sobreposições, escolhendo assim as formas mais representativas dos dados por meio de combinações lineares das variáveis originais (RINGNÉR, 2008).

3. MATERIAIS E MÉTODOS

3.1. PESQUISA DE CAMPO

Devido à complexidade e especificidade em relação ao processo de classificação de grãos, para um melhor entendimento deste processo aplicado em situações reais, foram feitas três visitas, in loco, em cooperativas da região. A primeira visita foi feita em Julho de 2013, em uma cooperativa de Guarapuava; a segunda foi em Novembro de 2013, em cooperativa de Ponta Grossa; e a terceira foi realizada em Março de 2014, na mesma cooperativa de Guarapuava. Nessas visitas, além de reuniões com os responsáveis e técnicos, foi feito um acompanhamento detalhado de todo o processo, desde a coleta e amostragem até a classificação dos grãos.

3.2. OBJETO DE ESTUDO

Foram utilizadas, no total, 2.000 amostras de grãos de milho, fornecidas por três cooperativas diferentes. As amostras foram selecionadas e classificadas pelos técnicos em cada uma das cooperativas. Os critérios seguidos pelas cooperativas para a classificação dos grãos, em sadios e ardidos, estão definidos nas instruções normativas (I.N.) MAPA nº 60/2011 e 18/2012 (CONAB, 2013).

3.3. EQUIPAMENTO USADO PARA DESEVOLVIMENTO DOS PROGRAMAS

O equipamento utilizado para a análise e extração de características de imagens, treinamento de algoritmos, validação e testes foi um Notebook ACER Aspire, Modelo 5733-6663, com Processador Intel Core i3 e 4GB de memória, com Sistema Operacional Linux UBUNTU versão 12.04 LTS.

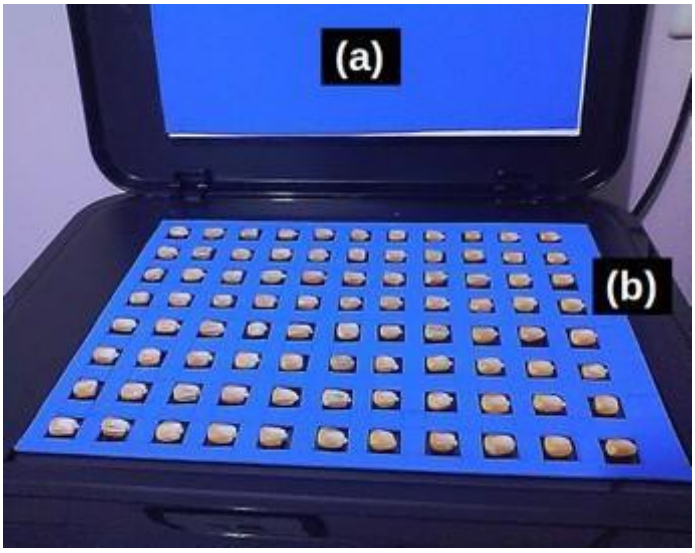
3.4. AQUISIÇÃO

A aquisição das imagens foi feita com um scanner SAMSUMG, Modelo SCX-4600 (Figura 15). O software utilizado para aquisição foi o XSane Imaging Scanner Program Versão 0.998, com tipo de arquivo: png; fundo plano; permitindo todas as cores; resolução 75/300/600 dpi; tamanho A4, e localização da área de digitalização no scanner: *x* superior esquerdo com valor de 0,5 cm, *y* superior esquerdo com o valor de 0,5 cm, *x* inferior direito com o valor de 21,0 cm, e *y* inferior direito com o valor de 29,0 cm. Para o fundo das imagens foi utilizado um EVA azul afixado ao

scanner (Figura 15(a)). O fundo azul foi o que mais favoreceu a detecção dos grãos entre os fundos testados (branco, preto, vermelho e azul), possivelmente porque a distância de cor entre a região do azul e aquelas do grão de milho são mais significativas.

Para o posicionamento dos grãos, foi utilizada uma matriz em EVA com 88 furos retangulares de 1,5 cm (largura) x 2,0 cm (altura) aproximadamente,Figura 15(b). Como muitos dos trabalhos correlacionados a esta pesquisa não mencionam a resolução utilizada, neste trabalho optou-se por trabalhar com as resoluções de 75 dpi, que corresponde a menor resolução do scanner; 300 dpi, que é uma resolução intermediária; e 600 dpi, que é a maior resolução do scanner. A escolha de três valores de resolução também foi para avaliar o impacto desta característica da imagem no resultado final.

Figura 15 – Scanner SAMSUNG usado para aquisição de imagens: (a) fundo em EVA; (b) matriz em EVA para posicionamento das sementes.



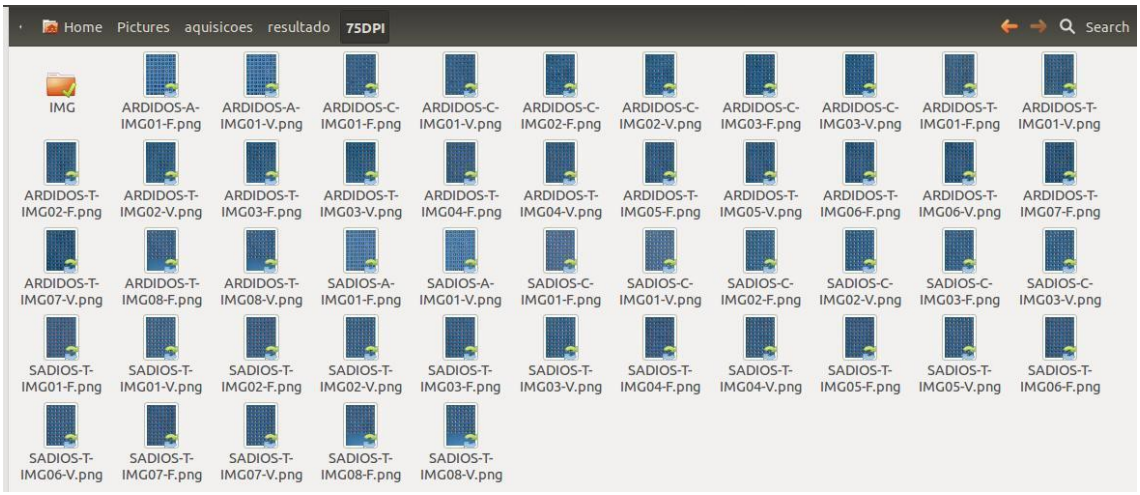
Fonte: Autor.

No momento da digitalização, a matriz de EVA, Figura 15(b), foi retirada, mantendo apenas os grãos posicionados. Para cada um dos grãos foram escaneados seus dois lados, identificados neste trabalho como frente e verso. Isso foi feito porque os grãos de milho possuem seus lados diferenciados e uma determinada característica (objeto de estudo), pode se manifestar em apenas um dos lados.

As imagens adquiridas com o scanner foram salvas em uma pasta correspondente à resolução, em dpi, da imagem. Sendo assim, uma imagem de 75 dpi foi salva em uma pasta chamada 75DPI (Figura 16); imagens com 300 dpi e 600

dpi de resolução foram salvas, respectivamente, em pastas chamadas 300DPI e 600DPI.

Figura 16 – Arquivos de imagens de aquisição com resolução de 75 dpi presentes na pasta 75DPI.



Fonte: Autor.

Ainda, como pode ser visto na Figura 16, as imagens foram salvas com o prefixo “IMG” seguido de um numerador sequencial de 2 dígitos, mais a literal “-F”, para frente, e “-V”, para verso; e a extensão do arquivo de imagem. A título de exemplificação, a primeira imagem se chama “IMG01-F.png” havendo também outra chamada “IMG01-V.png”.

Após o processo de segmentação, as imagens individuais de cada grão foram geradas em uma subpasta chamada IMG com o prefixo “IMG”, seguido do numerador sequencial de dois dígitos da imagem original adquirida, mais o prefixo “–”, seguido de um numerador sequencial de cada imagem. Esse número sequencial, neste trabalho, foi de 1 a 88 e representa, individualmente, cada um dos grãos posicionados no scanner em cada processo de digitalização. Dessa forma, os arquivos “IMG01-F.png” e “IMG01-V.png” geraram as imagens nomeadas de “IMG01-1.png” a “IMG01-88.png”.

3.5. SEGMENTAÇÃO E EXTRAÇÃO DE CARACTERÍSTICAS

Após a aquisição das imagens, todo o processo de pré-processamento, segmentação e extração de características foi realizado por dois softwares desenvolvidos para este propósito: o SEGMENTADOR (Apêndice A) e o EXTRATOR (Apêndice B). Para o desenvolvimento desses programas foi utilizada a linguagem de programação *Python Versão 2.7.3*, com o *Framework SimpleCV*

Versão 1.3.0 (PYTHON SOFTWARE FOUNDATION, 2012; DEMAAGD et al., 2012). Como ferramenta auxiliar para escrita, teste e depuração dos códigos foi utilizado o *Editor Python de Stani* (*Stani's Python Editor* - SPE), que é uma *IDE Python* escrita em Python disponível em <http://pythonide.stani.be/>.

O processo de pré-processamento e segmentação foi realizado pelo programa SEGMENTADOR. Esse programa foi responsável por: (i) fazer a leitura das imagens de frente e verso dos grãos, geradas na etapa de aquisição; (ii) correlacionar as imagens de frente e verso correspondentes; (iii) detectar e segmentar os grãos presentes na imagem, sendo eles sadios ou ardidos, usando filtro da mediana (para eliminação de ruidos, seção 2.2.2) e cálculo da distância de cor entre fundo e objeto de interesse (seção 2.2.5); (iv) unir em uma única imagem a imagem de cada objeto segmentado, correspondente a frente e o verso de cada grão; (v) salvar, em diretório previamente informado, as imagens finais geradas, contendo a frente e o verso de cada grão. Como entradas para o programa SEGMENTADOR foram fornecidas informações sobre a localização das imagens e resolução, para que o processo de segmentação fosse realizado com sucesso.

O SEGMENTADOR utilizou uma classe principal chamada Img para o processamento das imagens (Figura 17).

Figura 17 – Representação UML da Classe Img desenvolvida.

Img		
+ files	:	array()
+ dpi	:	string
+ origem	:	string
+ destino	:	string
+ color	:	array()
+ genereation()	:	void

Fonte: Autor

Após a execução do programa, as imagens resultantes foram geradas em uma subpasta chamada IMG.

Depois do processo de pré-processamento e segmentação, foi realizada a extração de características, com auxílio do software EXTRATOR. Esse programa foi responsável por: (i) ler todas as imagens de grãos geradas; (ii) extrair os histogramas de cada imagem no espaço de cor previamente informado (sendo que no espaço de cor HSV foi utilizado o valor de H); (iii) extrair as características de

textura com base no padrão LBP; (iv) organizar as informações extraídas e gerar o arquivo com as características coletadas das imagens no padrão do Weka. Para a geração desse arquivo, no padrão Weka o programa EXTRATOR faz uso de uma Classe denominada Arff (Figura 18).

Figura 18 – Representação UML da Classe Arff desenvolvida.

Arff		
+ relation	:	string
+ atributes	:	array()
+ datas	:	array()
+ addAtribute(string)	:	void
+ addData(string)	:	void
+ genereation(string)	:	void

Fonte: Autor.

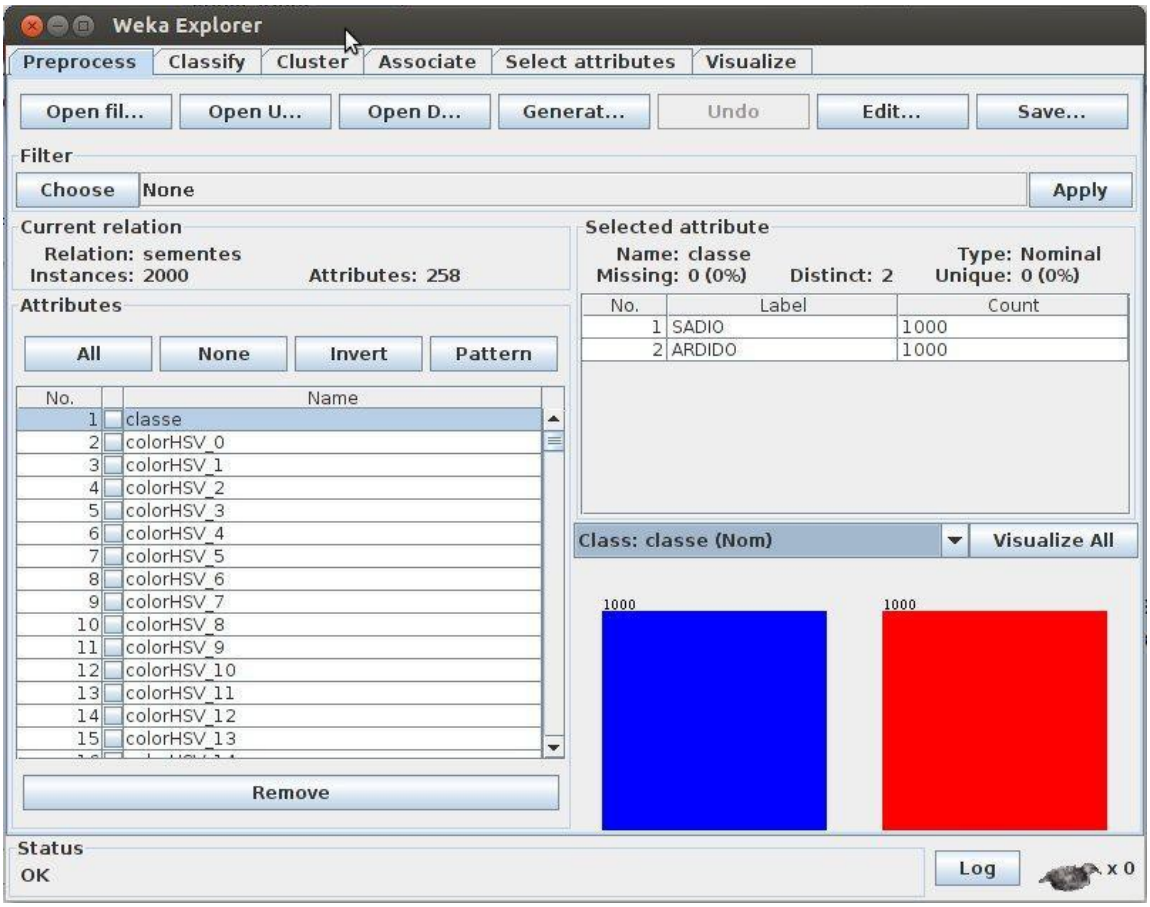
Para o processo de extração de características foi informado o diretório com os arquivos das imagens, a resolução em dpi (75/300/600), os dados para extração, e o nome do arquivo ARFF a ser gerado no formato padrão usado pelo WEKA. Com relação aos dados para extração, as opções individuais ou combinadas disponibilizadas pelo programa são: canal R, canal G, canal B, HSV (canal H) e LBP.

3.6. MINERAÇÃO DE DADOS

Foram geradas combinações de bases de dados com o Software EXTRATOR, que foram submetidas ao processo de mineração de dados com o Weka. As combinações relacionaram resolução (DPI) x Tipo (Cor/Textura).

Os arquivos gerados pelo Software EXTRATOR foram utilizados como base de dados de entrada, para o processo de mineração de dados, na ferramenta Weka. Cada base, individualmente, foi selecionada dentro da ferramenta e carregada no Weka Explorer, na guia Preprocess, para pré-processamento, e onde foram identificados os atributos, classe e dados (Figura 19).

Figura 19 – Carregamento de base de dados no Weka Explorer para pré-processamento.

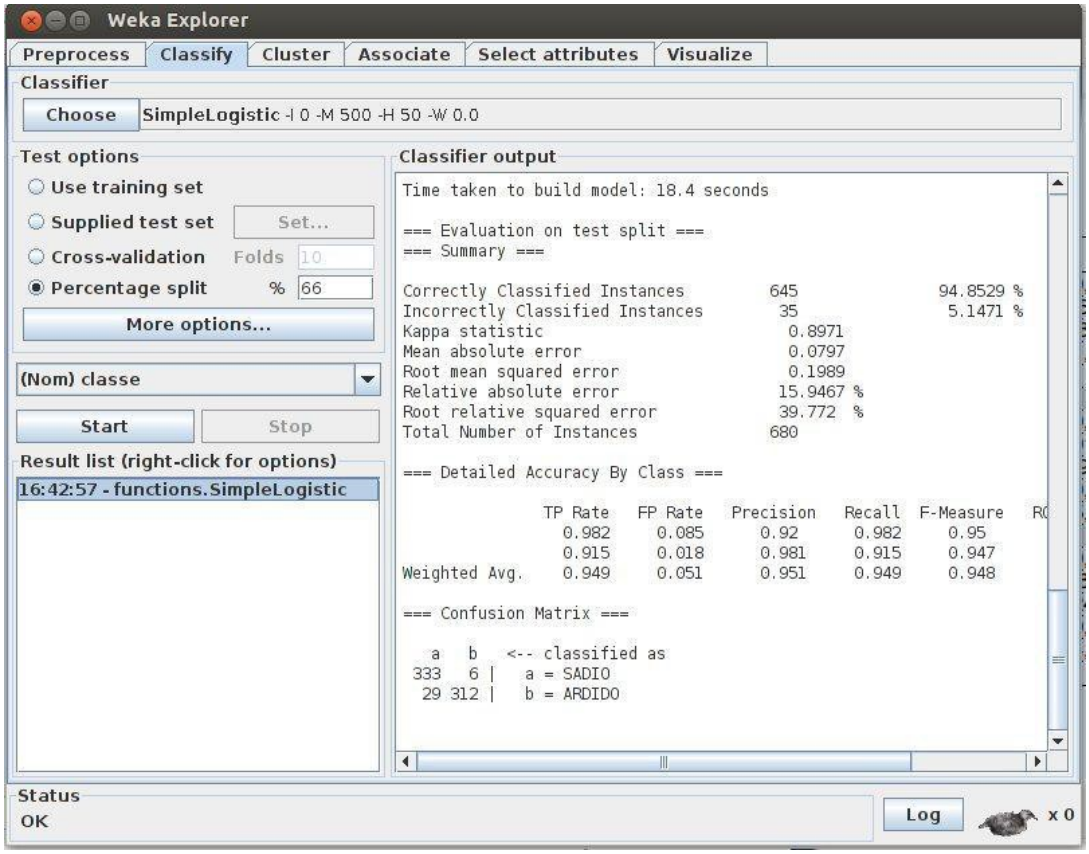


Fonte: Autor.

3.6.1 Método Divisão de Amostras

Após o carregamento da base e seu pré-processamento, os dados foram submetidos ao processo de classificação. Isso foi feito na guia *Classify* (Classificação), onde foi selecionado o algoritmo classificador (*Classifier*) que se deseja executar, o método a ser aplicado (*Test options*), que no caso deste trabalho foi o *Percentage split* (divisão percentual) de 66% (método *Holdout* ou Divisão de Amostras), e o campo de classificação (Figura 20). Todos os algoritmos disponibilizados pela ferramenta foram executados.

Figura 20 – Execução do algoritmo de mineração de dados na guia *Classify* do Weka Explorer.



Fonte: Autor

Após a execução do algoritmo foram selecionados e então coletados alguns dos dados de interesse que são apresentados na janela *Classifier Output* (Resultado da Classificação). As informações de interesse escolhidas para análise neste trabalho foram: *correctly classified* (classificadas corretamente), *relative absolute error* (erro relativo absoluto), *kappa* e *time taken* (tempo de processamento). Esses parâmetros foram escolhidos por permitirem avaliar o grau de acerto na classificação e o desempenho de cada algoritmo executado, além de serem aqueles que geralmente são apresentados em trabalhos que realizam análises com mineração de dados.

3.6.2 Método de Validação Cruzada

Para o método de Validação Cruzada foram geradas 21 bases de dados referentes às combinações de cor, textura e dpi (Tabela 2). Cada base de dados foi gerada com informações de 1.000 amostras de grãos, composta por 500 grãos sadios e 500 grãos ardidos.

TABELA 2 – Bases de dados geradas para o método de validação cruzada.

Cor/Textura	DPI 75 300 600
R	X X X
G	X X X
B	X X X
RGB	X X X
HSV	X X X
LBP	X X X
HSV-LBP	X X X

As tabelas geradas foram carregadas no Weka, usando o Módulo *Experimenter.* Na tela desse módulo foram definidos o arquivo de saída e os algoritmos para execução. Foi definida também a execução do método de Validação Cruzada com a opção de 10 *folds*.

3.6.3 TESTE E VALIDAÇÃO

O Weka gera um modelo para cada método executado e, para efeito de teste do modelo proposto, foi utilizado o programa TESTMODEL (Apêndice C). O modelo escolhido para ser elaborado e testado com o TESTMODEL foi com base nos melhores resultados para o método de Validação Cruzada, simplicidade de implementação e eficiência.

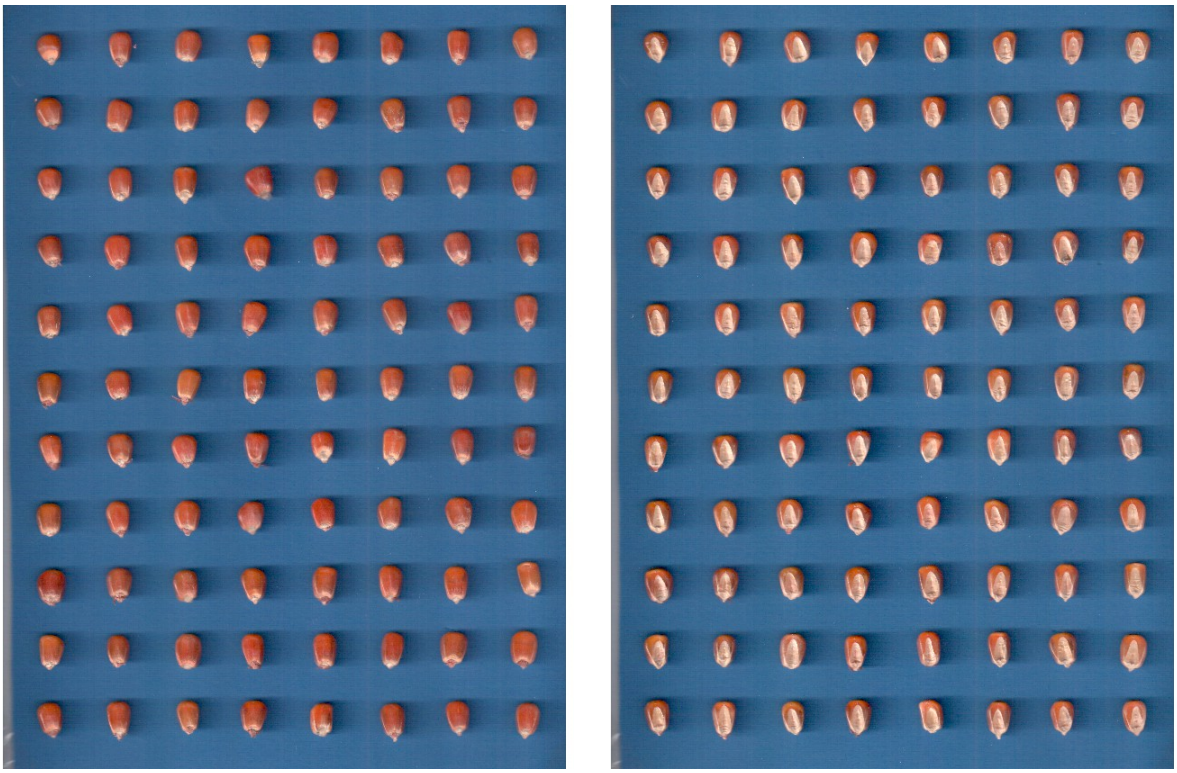
Para teste e validação do modelo foram selecionadas 500 imagens novas (não utilizadas no processo de Validação Cruzada), sendo 495 imagens de grãos sadios (99%) e 5 imagens de grãos ardidos (1%). Essa proporção de 1% para grãos ardidos foi baseada na Instrução Normativa (I.N.) MAPA nº 60/2011 (CONAB, 2013).

4. RESULTADOS E DISCUSSÕES

4.1. AQUISIÇÃO

No processo de aquisição de imagens foram obtidas imagens de todos os grãos fornecidos pelas três cooperativas e, como resultado, foram geradas, no total, 144 imagens. A Figura 21 apresenta um exemplo de duas imagens geradas na aquisição, uma da frente, Figura 21(a), e outra do verso, Figura 21(b), para um grupo de grãos sadios.

Figura 21 – Imagens (a) frente e (b) verso de grãos sadios, com uma resolução de 75 dpi, geradas na etapa de aquisição.



(a) (b)

Fonte: Autor.

Na Tabela 3 está o relatório das 144 imagens geradas, sendo 48 imagens para cada resolução (75/300/600 dpi), metade delas correspondendo à frente e a outra metade ao verso.

TABELA 3 – Resultado da aquisição das imagens.

Cooperativa	Grãos Sadios (a)	Grãos Ardidos (b)	Imagens Frente/dpi * (c)	Imagens Verso/dpi ** (d)	Total de Imagens ***
A	88	88	2	2	12
B	264	264	6	6	36
Ca	616	616	14	14	84
Cb	32	32	2	2	12
TOTAL	**1000**	**1000**	**24**	**24**	**144**

* Imagens frente = **(a+b)/(88 ou 32)** **** Images verso = (a+b)/(88 ou 32)** ***** Total de Images = (c+d)*3**

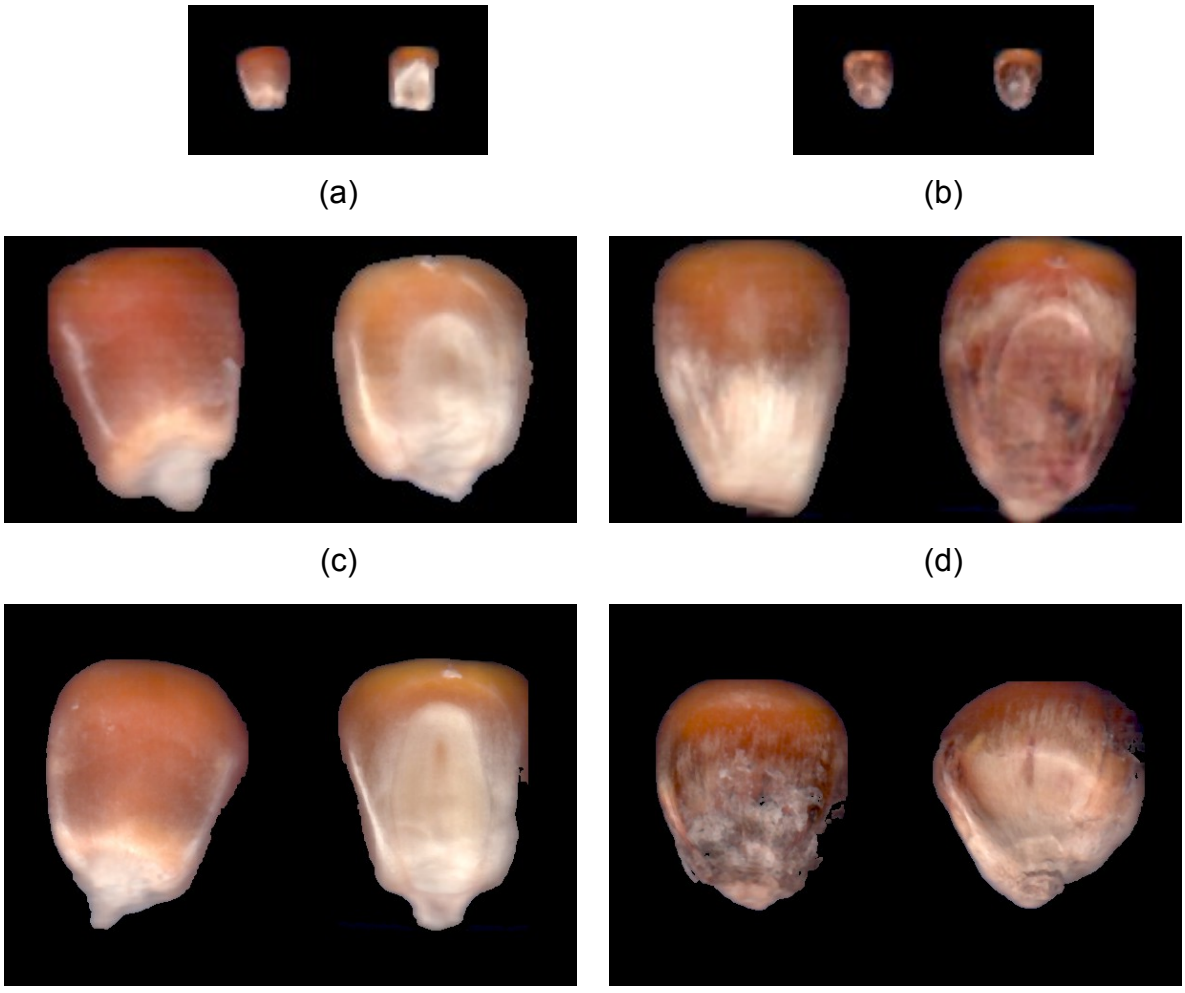
No caso da cooperativa C, conforme pode ser verificado na Tabela 3, foi necessário subdividir o processo de aquisição em Ca e Cb, porque o total de grãos fornecidos não foi um múltiplo de 88, o número máximo de grãos por aquisição, o que gerou uma última aquisição com 32 grãos.

4.2. PRÉ-PROCESSAMENTO, SEGMENTAÇÃO E EXTRAÇÃO DE CARACTERÍSTICAS

Com o programa SEGMENTADOR foram executadas as etapas de pré-processamento e segmentação. Ao final do processamento de todas as imagens adquiridas, foram geradas 5990 imagens de grãos, sendo 1990 para a resolução de 75 dpi, 2000 para a resolução de 300 dpi e 2000 para a resolução de 600 dpi. Para a filtro da mediana, foi adotado o tamanho padão da máscara do framework SimpleCV (3 x 3); e na etapa da segmentação, no cálculo da distância de cor, foi informado para um dos parâmetros a cor de fundo "azul", cujos valores estão também estão pré-definidos neste framework. Da diferença entre os valores de imagens geradas após o processo de segmentação (do total de 2.000 imagens de grãos referente a 1.000 de grãos sadios e 1.000 ardidos, e foram obtidas somente 1990 imagens), afirma-se que a resolução de 75 dpi apresentou problemas nesta etapa de processamento. Essa diferença não ocorreu para imagens com as resoluções de 300 dpi e 600 dpi.

A Figura 22 apresenta um exemplo de cada imagem obtida para o grão sadio e ardido, após o pré-processamento e a segmentação, para as três resoluções escolhidas.

Figura 22 – Imagens geradas após o pré-processamento e a segmentação: (a) grão sadio, 75 dpi; (b) grão ardido, 75 dpi; (c) grão sadio, 300 dpi; (d) grão ardido, 300 dpi; (e) grão sadio, 600 dpi e (f) grão ardido, 600 dpi.



(a) (b)

(c) (d)

(e) (f)

Fonte: Autor

Realizando uma comparação visual entre a Figura 22(a), Figura 22(c) e Figura 22(e) é possível perceber que quanto maior a resolução maior a quantidade de detalhes presentes na imagem, que esta relacionada à quantidade de pixels capturados. Entretanto, maior quantidade de pixels na imagem implica em maior tamanho do arquivo e maior custo computacional.

O programa SEGMENTADOR também foi responsável pela extração de características das imagens. Foram extraídos os histogramas de cada imagem e sua classe e gerados os arquivos no padrão Weka, com a extensão ARFF. Na Figura 23

é apresentado um exemplo de um arquivo gerado com as informações de todas as imagens com resolução de 75 dpi e para o canal B do espaço de cor RGB.

Figura 23 – Conteúdo de arquivo no padrão Weka (ARFF).

```
@relation COLOUR-B-75
@attribute colorB_0 real
@attribute colorB_1 real
@attribute colorB_2 real
@attribute colorB_3 real
...
@attribute colorB_253 real
@attribute colorB_254 real
@attribute colorB_255 real
@attribute classe { SADIO, ARDIDO}
@data
9640.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,22.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,40.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,119.0,0.0,0.0,0.0,0.0,0
,0.0,0.0,0.0,0.0,0.0,0.0,100.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,112.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,10
7.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,117.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,86.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,65.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,56.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,61.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,52.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,49.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,72.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,127.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,147.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,151.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,67.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,38.0,0.0,0.0,0.0,0.0,0.0,
0,0.0,0.0,0.0,0.0,0.0,0.0,17.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,4.0,SADIO
9610.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,12.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,12.0,0.0,0.0,0.0,0.0,0.0,0.0,
0,0.0,0.0,0.0,0.0,0.0,18.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,84.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,70.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,155.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,137.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,145.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,114.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,88.0,0.0,0.0,0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,121.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,104.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0,0.0,0.0,66.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,81.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,68.0,0.0,0.0,0.0,0.0,
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,65.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,125.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0,76.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,37.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,37.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0,0.0,0.0,0.0,0.0,0.0,0.0,20.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,5.0,SADIO
9764.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,29.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,84.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,121.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,160.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,108.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,115.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,82.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,89.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,85.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,68.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,87.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,98.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,112.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,82.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,49.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,52.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,28.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,30.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,7.0,SADIO
```

Fonte: Autor.

Na primeira linha do arquivo é informado seu nome com o parâmetro @relation onde COLOUR-B-75 é o nome do arquivo ou base gerada. Com o parâmetro @attribute nas linhas seguintes são informados os campos de dados que serão gerados na base. Após o parâmetro @data são geradas as linhas de cada imagem, com os valores dos campos informados, na ordem em que foram informados e separados por vírgula. Esses valores correspondem ao histograma de cada imagem com a quantidade de pixels para cada canal de cor que vai de 0 a 255. O último valor da linha é a classificação da imagem.

No total foram gerados, pelo SEGMENTADOR, 22 arquivos com as

características extraídas das imagens pelo SEGMENTADOR (Figura 24).

Figura 24 – Arquivos padrão Weka (ARFF) gerados pelo SEGMENTADOR.

```
base-COLOUR-B-75dpi.arff
base-COLOUR-B-300dpi.arff
base-COLOUR-B-600dpi.arff
base-COLOUR-R-75dpi.arff
base-COLOUR-R-300dpi.arff
base-COLOUR-R-600dpi.arff
base-COLOUR-G-75dpi.arff
base-COLOUR-G-300dpi.arff
base-COLOUR-G-600dpi.arff
base-COLOUR-R-G-B-75dpi.arff
base-COLOUR-R-G-B-300dpi.arff
base-COLOUR-R-G-B-600dpi.arff
base-COLOUR-HSV-75dpi.arff
base-COLOUR-HSV-300dpi.arff
base-COLOUR-HSV-600dpi.arff
base-COLOUR-HSV-LBP-75dpi.arff
base-COLOUR-LBP-75dpi.arff
base-COLOUR-LBP-300dpi.arff
base-COLOUR-LBP-600dpi.arff
base-COLOUR-HSV-LBP-300dpi.arff
base-COLOUR-HSV-LBP-600dpi.arff
```

Fonte: Autor

Conforme pode ser observado na Figura 24, todos os arquivos foram gerados com sua nomenclatura contendo informações sobre qual espaço de cor, textura e resolução da imagem.

4.3. MINERAÇÃO DE DADOS - MÉTODO DA DIVISÃO DE AMOSTRAS

Nas Tabelas 4 a 6 são apresentados os resultados das melhores combinações DPI x Tipo, tendo como referência os trabalhos Steenhoek et al. (2001), Dragavova et al. (2010) e Potter et al. (2012).

Na Tabela 4 estão os resultados da mineração de dados para imagens com diferentes resoluções e com base no histograma LBP. Na Tabela 5 estão os resultados da mineração de dados para imagens com diferentes resoluções e com base no histograma HSV (canal H) e na Tabela 6 são fornecidos os valores obtidos da mineração de dados para imagens com diferentes resoluções e com base no histograma do canal B (azul) do espaço de cor RBG.

TABELA 4 – Resultado da Mineração de Dados com base no histograma LBP.

Algoritmo	DPI	Classificação correta (%)	Erro relativo absoluto (%)	Kappa	Tempo de processamento (s)
Dagging	75	96,02	16,08	0,92	0,56
LMT	“	95,58	13,94	0,91	63,74
SimpleLogistic	“	95,58	13,94	0,91	15,89
SMO	“	95,14	9,70	0,90	1,15
FT	“	94,85	10,41	0,89	2,52
Dagging	300	97,20	11,26	0,94	0,56
LMT	“	96,91	10,16	0,93	57,98
SimpleLogistic	“	96,91	10,16	0,93	16,91
SMO	“	96,17	7,64	0,92	0,61
FT	“	96,32	8,31	0,92	2,89
Dagging	600	94,11	15,41	0,88	1,01
LMT	“	96,17	12,01	0,92	83,69
SimpleLogistic	“	96,17	12,01	0,92	15,71
SMO	“	95,58	8,82	0,91	1,24
FT	“	95,44	9,20	0,90	3,00

Na Tabela 4 observa-se que a concentração das melhores acurácias foi obtida para a resolução de 300 dpi, sendo que o algoritmo Dagging foi o melhor deles, com uma porcentagem de classificação correta de 97,20%, e com um tempo de processamento de 0,56 segundos. Ainda para essa resolução, embora os algoritmos LMT e SimpleLogistic estejam com os mesmos valores de classificação correta, erro relativo absoluto e Kappa, o tempo de processamento é um fator de diferenciação. Se for considerado apenas o erro relativo absoluto e o tempo de processamento, destaca-se o algoritmo SMO, para a resolução de 300 dpi, que teve o menor percentual em relação ao erro, com um tempo de processamento similar ao algoritmo de melhor acurácia (Dagging).

TABELA 5 – Resultado da Mineração de Dados com base no histograma HSV.

Algoritmo	DPI	Classificação correta (%)	Erro relativo absoluto (%)	Kappa	Tempo de processamento (s)
Dagging	75	85,29	44,73	0,70	0,94
LMT	“	85,73	39,88	0,71	69,41
SimpleLogistic	“	85,73	39,88	0,71	12,66
SMO	“	84,70	30,58	0,69	2,35
FT	“	82,94	34,18	0,65	3,23
Dagging	300	94,41	13,17	0,88	0,23
LMT	“	94,85	15,94	0,89	79,45
SimpleLogistic	“	94,85	15,94	0,89	21,55
SMO	“	95,73	8,52	0,91	0,41
FT	“	95,44	10,53	0,90	3,31
Dagging	600	95,58	10,47	0,91	0,51
LMT	“	95,73	13,01	0,91	80,15
SimpleLogistic	“	95,73	13,01	0,91	20,42
SMO	“	96,02	7,94	0,92	0,24
FT	“	96,17	8,99	0,92	2,46

Com relação à Tabela 5, a concentração das melhores classificações corretas foi obtida para as imagens com 600 dpi, e o algoritmo FT foi o que apresentou o melhor valor de classificação. Entretanto, quando considerados outros fatores, como o erro relativo absoluto e o tempo de processamento, destaca-se, novamente, o algoritmo SMO, para a resolução de 600 dpi, com os menores valores para o erro relativo absoluto (7,94 %) e para a o tempo de processamento (0,41 s) com relação aos demais algoritmos analisados.

Nota-se que a Tabela 4 e Tabela 5 apresentaram os melhores resultados com as imagens com resolução de 300 dpi e de 600 dpi, respectivamente. Uma explicação poderia ser o fato de que os resultados apresentados na Tabela 4 utilizaram dados de textura, como entrada para o processo de mineração de dados, enquanto que os resultados da Tabela 5 utilizaram informações do espaço de cor

HSV (canal H) das imagens. Sendo assim, durante o processo de extração de características, a textura foi favorecida com a resolução média, enquanto que para o espaço de cor HSV houve um favorecimento com a resolução superior, 600 dpi.

TABELA 6 – Resultado da Mineração de Dados com base no histograma do canal B (azul) do espaço de cor RGB.

Algoritmo	DPI	Classificação correta (%)	Erro relativo absoluto (%)	Kappa	Tempo de processamento (s)
Dagging	75	92,50	18,88	0,85	0,32
LMT	“	94,55	19,65	0,89	123,09
SimpleLogistic	“	94,55	19,65	0,89	21,35
SMO	“	94,11	11,76	0,88	1,8
FT	“	93,38	14,86	0,86	2,85
Dagging	300	91,47	17,97	0,82	0,68
LMT	“	95,44	18,62	0,90	140,36
SimpleLogistic	“	95,44	18,62	0,90	20,48
SMO	“	94,41	11,17	0,88	1,45
FT	“	93,67	13,22	0,87	2,66
Dagging	600	93,38	18,58	0,86	0,34
LMT	“	93,82	21,39	0,87	113,51
SimpleLogistic	“	93,82	21,39	0,87	19,74
SMO	“	93,23	13,52	0,86	1,49
FT	“	93,08	15,70	0,86	2,8

No caso da Tabela 6, onde foi utilizado o canal B do espaço de cor RGB, observa-se que houve uma concentração de melhores resultados para a classificação correta nas resoluções mais baixas, 75 dpi e 300 dpi. A resolução de 300 dpi foi ainda o que possibilitou melhores resultados, com os algoritmos LMT e SimpleLogistic apresentando os mesmos valores de classificação correta, erro relativo absoluto e Kappa, e o tempo de processamento sendo, novamente, o fator de diferenciação. Para a resolução de 600 dpi, os algoritmos tiveram um desempenho de classificação correta mais homogêneo, variando entre 93,08 % e 93,82 %.

Comparando com as outras tabelas (Tabela 4 e Tabela 5), observa-se que o tempo de processamento dos algoritmos, de uma maneira geral, foi maior,

principalmente para o algoritmo LMT, nas três resoluções. Analisando ainda as três as tabelas, pode-se perceber que LBP (Tabela 4) e HSV (Tabela 5) apresentaram os melhores resultados em termos de classificação correta. Em função disso, foi gerada uma combinação com ambos os parâmetros, o LBP e o HSV, com a expectativa de melhorar os resultados apresentados até o momento. Os resultados dessa combinação está na Tabela 7.

TABELA 7 – Resultado da Mineração de Dados com base no histograma HSV x LBP.

Algoritmo	DPI	Classificação correta (%)	Erro relativo absoluto (%)	Kappa	Tempo de processamento (s)
Dagging	75	95,29	14,73	0,90	0,64
LMT	“	95,14	15,34	0,90	152,7
SimpleLogistic	“	95,14	15,34	0,90	28,98
SMO	“	94,70	10,58	0,89	1,81
FT	“	94,41	11,33	0,88	6
Dagging	300	97,35	6,91	0,94	0,74
LMT	“	98,23	7,36	0,96	101,76
SimpleLogistic	“	98,23	7,36	0,96	29,94
SMO	“	97,94	4,11	0,95	0,99
FT	“	97,64	6,50	0,95	5,53
Dagging	600	96,76	7,79	0,93	0,33
LMT	“	97,20	10,22	0,94	98,39
SimpleLogistic	“	97,20	10,22	0,94	24,6
SMO	“	97,35	5,29	0,94	0,82
FT	“	97,35	6,28	0,94	4,57

Considerando somente o percentual de classificação correta, os algoritmos avaliados para imagens com resolução de 300 dpi e 600 dpi foram os que concentraram os melhores resultados. Para a resolução de 300 dpi, destacaram-se os algoritmos LMT e SimpleLogistic, com uma porcentagem de classificação correta de 98,23%. Entretanto, os valores de erro relativo absoluto foram de 7,36 %, para ambos, e o de tempo de processamento foi de 101,96 s e 29,94 s, respectivamente. Vale ainda destacar que o Algoritmo SMO, para resolução de 300 dpi, além de um percentual de classificação correta muito próximo aos dois anteriores, conseguiu uma taxa de erro bem menor (4,11 %) e com tempo de processamento de 0,99 s.

Ao se comparar as Tabelas 4, 5 e 7, verifica-se que a combinação dos

parâmetros de cores HSV (canal H) junto com os de textura LBP foi uma boa estratégia para a melhoria dos resultados. Essa estratégia foi utilizada em função das características que distinguem grãos sadios dos grãos ardidos, conforme descrito por Pinto (2005). Draganova et al. (2010a) também usou este tipo de combinação para melhorar a acurácia final no processo de classificação.

Todos os resultados apresentados nas Tabelas 4 a 7 têm seus índices de classificação correta acima de 90%. Para situações em que a análise irá se basear em outras informações adicionais como a menor taxa de erro, então, um bom algoritmo seria o SMO, para imagens com resolução de 300 dpi, com classificação de 97,94%, baseado em dados de textura (LBP) e espaço de cor (HSV, canal H). Para situações onde se busca um algoritmo com menor tempo de processamento, tem-se o Dagging, para imagens com resolução de 300 dpi, com 0,23 s de tempo de processamento, 94,41% de classificação correta e 13,17 % de erro relativo absoluto, para o espaço de cor HSV.

As demais análises, com resultados de classificação correta menores que as apresentadas, podem ser vistas nos arquivos disponibilizados no Google Drive, conforme mencionado no início desta seção.

Vale ressalvar que os algoritmos elencados tiveram os desempenhos listados para a base de dados estudada. Rendimentos diferentes podem ser obtidos para outras bases de dados, já que os algoritmos têm seus desempenhos diretamente ligados às características da base de dados em análise (CAMILO & SILVA, 2009).

4.4. MINERAÇÃO DE DADOS - MÉTODO DA VALIDAÇÃO CRUZADA

Os resultados de todos os testes da validação cruzada estão disponibilizados no Google Drive, conforme mencionado no início da seção 4.3. Como pode ser verificado na seção 4.3, vários algoritmos obtiveram resultados igual/superiores a 90%, em variadas combinações de cor, textura e dpi, utilizando o método da Divisão de Amostras. Na Tabela 8 estão relacionados os algoritmos que obtiveram taxa de acerto igual ou superior a 99% para o método de Validação Cruzada.

TABELA 8 – Algoritmos com taxa de acerto igual ou superior a 99% para o método de Validação Cruzada.

Cor/Textura	DPI	Grupo	Algoritmo	Classificação correta (%)	Erro relativo absoluto (%)	Kappa	*Tempo de processamento* (s)
HSV-LBP	300	functions	SMO	99.90	0.20	1.00	0.15
HSV-LBP	300	meta	RotationForest	99.80	9.27	1.00	22.26
HSV-LBP	300	functions	SimpleLogistic	99.70	1.76	0.99	10.80
HSV-LBP	300	functions	Logistic	99.60	0.82	0.99	0.39
HSV-LBP	300	meta	MultiClassClassifier	99.60	0.82	0.99	0.38
HSV-LBP	300	functions	SPegasos	99.50	1.00	0.99	1.86
HSV-LBP	300	trees	FT	99.50	2.96	0.99	1.62
HSV-LBP	300	lazy	IB1	99.40	1.20	0.99	4.66
HSV-LBP	300	lazy	IBk	99.40	1.42	0.99	3.24
HSV-LBP	300	meta	RandomCommittee	99.30	16.54	0.99	0.36
HSV-LBP	600	functions	SimpleLogistic	99.30	3.15	0.99	13.00
HSV-LBP	600	functions	SMO	99.30	1.40	0.99	0.17
HSV	600	functions	SPegasos	99.20	1.60	0.98	0.84
HSV	600	lazy	IB1	99.20	1.60	0.98	2.22
HSV	600	lazy	IBk	99.20	1.82	0.98	0.57
HSV-LBP	600	functions	SPegasos	99.20	1.60	0.98	1.84
HSV-LBP	600	meta	RandomCommittee	99.20	7.96	0.98	0.32
HSV-LBP	600	meta	RotationForest	99.20	5.65	0.98	22.53
HSV-LBP	600	trees	FT	99.20	3.44	0.98	1.31
HSV	600	functions	SMO	99.10	1.80	0.98	0.07
HSV	600	functions	SimpleLogistic	99.00	3.46	0.98	5.03
HSV-LBP	300	meta	ThresholdSelector	99.00	10.67	0.98	0.72
HSV-LBP	600	functions	Logistic	99.00	1.90	0.98	0.90
HSV-LBP	600	meta	MultiClassClassifier	99.00	1.90	0.98	2.17

Pode-se observar na Tabela 8 que os melhores resultados ficaram concentrados entre resoluções de 300 dpi e 600 dpi, para o espaço de cor HSV (canal H), e, na maioria, com a combinação da característica de textura no padrão LBP. Observa-se que o percentual de classificação correta entre os algoritmos selecionados estão próximos, porém existem diferenças maiores entre os algoritmos para os valores de erro relativo absoluto e tempo de processamento. Trabalhos que busquem uma diferenciação entre os mesmos poderão usar esses dados para este propósito, se for o caso.

4.5. IMPLEMENTAÇÃO DO MODELO E TESTE

Na Figura 25 é apresentado o modelo matemático gerado para algoritmo SimpleLogistic quando o espaço de cor HSV (canal H), textura pelo padrão LBP, e resolução de 300 dpi são usados. Esse modelo foi escolhido por ser simplificado, de fácil implementação e por necessitar de poucos parâmetros de cálculo (seção 3.6.3).

Figura 25 – Modelo matemático para implementação do algortimo SimpleLogistic.

```
=== Classifier model (full training set) ===
SimpleLogistic:

Class 0 :                                   Class 1 :
-1.86 +                                     1.86 +
[colorHSV_205] * -0.09 +                    [colorHSV_205] * 0.09  +
[colorHSV_217] * -0.08 +                    [colorHSV_217] * 0.08  +
[colorHSV_220] * -0.07 +                    [colorHSV_220] * 0.07  +
[colorHSV_230] * -0.04 +                    [colorHSV_230] * 0.04  +
[colorLBP_30]  * 0     +                    [colorLBP_30]  * 0     +
[colorLBP_50]  * -0.07 +                    [colorLBP_50]  * 0.07  +
[colorLBP_54]  * -0.03 +                    [colorLBP_54]  * 0.03  +
[colorLBP_66]  * 0.06  +                    [colorLBP_66]  * -0.06 +
[colorLBP_124] * 0     +                    [colorLBP_124] * 0     +
[colorLBP_125] * 0.03  +                    [colorLBP_125] * -0.03 +
[colorLBP_127] * 0     +                    [colorLBP_127] * 0     +
[colorLBP_131] * 0     +                    [colorLBP_131] * 0     +
[colorLBP_193] * 0     +                    [colorLBP_193] * 0     +
[colorLBP_216] * -0.03 +                    [colorLBP_216] * 0.03  +
[colorLBP_240] * 0                          [colorLBP_240] * 0
```

Fonte: Autor

O modelo apresentado na Figura 25 contém a equação de classificação para cada uma das classes, sendo a classe 0 correspondente aos grãos sadios e a classe 1 aos grãos ardidos. Como pode ser observado trata-se de um modelo de simples implementação e que necessita de poucos parâmetros para o seu cálculo. O modelo em questão utiliza apenas 4 parâmetros de cores dos 256 disponíveis no histograma da imagem para o padrão de cor HSV e 11 parâmetros de textura entre os 256 valores de textura disponíveis no histograma da imagem.

A Tabela 9 apresenta os resultados obtidos com a implementação do modelo SimpleLogistic.

TABELA 9 – Resultado da implementação do modelo SimpleLogistic.

Tipo	Quantidade	Acerto	Erro	Classificação Correta
Sadios	495	494	1	99,80%
Ardidos	5	5	0	100,00%
Total	**500**	**499**	**1**	**99,80%**

Conforme pode ser observado na Tabela 9, a implementação do modelo para o algoritmo SimpleLogistic, conforme Figura 25, comprovou os resultados estatísticos apresentados na Tabela 8, referente ao algoritmo selecionado com uma pequena variação percentual de 0,10% a mais de precisão.

4.6. DISCUSSÃO DOS RESULTADOS

Analisando os trabalhos relacionados (Steenhoek et al., 2001; Draganova et al., 2010a; Draganova et al., 2010b; Potter et al., 2012), observa-se que os mesmos se concentraram no uso dos algoritmos PNN, Fuzzy e PCA. Entretanto, conforme pode ser observado na Tabela 10, uma tabela comparativa dos métodos computacionais, dos parâmetros de imagens utilizados e os respectivos resultados, apenas o uso do algoritmo não determina uma melhor acurácia no resultado final.

TABELA 10 – Comparação deste trabalho com os correlatos.

Descrição	Steenhoek et al. (2001)	Draganova et al. (2010a)	Draganova et al. (2010b)	Potter et al. (2012)	Nesse Trabalho
Algoritmo	PNN	PNN, Fuzzy	PCA	PCA	SimpleLogistic
Espaço de Cor	RGB	HSV, Lab, XYZ, xyY e YCbCr	Variação onda espectral	RGB,HSV e CIELAB	RGB,HSV
Melhor resultado	RGB	Combinados	-	HSV	HSV
Textura	-	Estatísticos	-	-	LBP
Software	NeuroShell EasyClassifyer	Matlab 7.1	Pirouette 3.11 SIMCA	Matlab	Weka
Aquisição	Camera Digital	Camera Digital	Espectometro	Camera Digital	Scanner
Imagens	720	200	300	400	2000
Taxa acerto	**93%**	**98%**	**99,3%**	**89%**	**99,8%**

Comparando-se os trabalhos verifica-se que outros fatores podem influenciar o resultado. Esses fatores podem estar relacionados com os métodos ou protocolos de aquisição, características utilizadas para o processo de segmentação, processamento e análise bem como a quantidade de amostras testadas. Draganova et al. (2010a) conseguiram um resultado melhor que o de Steenhoek et al. (2001), e isto pode estar relacionado aos espaços de cores utilizados em conjunto com dados de texturas. Entretanto, ressalta-se que Draganova et al. (2010a) usaram uma quantidade de imagens significativamente menor que Steenhoek et al. (2001), sendo assim não é possível determinar o quanto isto poderia influenciar no resultado.

Draganova et al. (2010a) em seu trabalho demonstrou que o espaço de cor RGB sozinho não apresenta bons resultados. Potter et al. (2012), mostraram que o espaço de cor HSV apresentou melhor resultado em comparação com o RGB. Isso também foi confirmado neste trabalho. Verificou-se também que, quando foi adicionada a textura (padrão LBP) no conjunto de dados contendo o espaço de cor HSV, foi possível melhorar os resultados, o que novamente está de acordo com o apresentado por Draganova et al. (2010a).

O método de Validação Cruzada vem sendo utilizado amplamente, pois sua estratégia permite uma melhor distribuição entre os dados de treinamento e teste (ARLOT et al., 2010). Em relação aos métodos de mineração de dados, para treinamento e teste, foi verificado que o método de Validação Cruzada apresentou resultados superiores ao método Divisão de Amostras, com taxas de acerto de entre 50% e 99,9% entre os algoritmos. Kohavil et al. (1995) apresentaram um trabalho em que foram comparados os métodos de treinamento e teste, e o pior resultado foi o do método Divisão de Amostras e o melhor com o de Validação Cruzada, apoiando os resultados conseguidos nesta pesquisa. Steenhoek et al. (2001) e Draganova et al. (2010a), aplicando as normas do Ministério da Agricultura de seus locais de trabalho como parâmetros para a realização do trabalho obtiveram uma taxa de acurácia de 93 % e 98 % (Tabela 10).

Diferentemente dos trabalhos apresentados na Tabela 10, que utilizaram apenas um único método ou algoritmo, este trabalho demonstra que é possível chegar a bons resultados, com acurácia superior a 99%, com diferentes algoritmos (Tabela 8). Deve-se destacar também o número significativo de grãos, 2000 grãos, usados nesta pesquisa que, comparado com o apresentado na Tabela 10, chega a ser de 3 a 10 vezes superior. Buscou-se trabalhar com esse número de grãos porque em situações reais a quantidade de grãos por amostras é significativa, e, com isto, quanto maior a variedade de informações sobre os grãos sadios e ardidos, melhor é a determinação do algoritmo mais indicado para atender às necessidades desta aplicação.

O resultado relacionado ao tempo de processamento é apresentado nas Tabelas 4 a 8, já que frequentes mudanças de parâmetros em relação a definição do objeto em si (grãos ardidos), podem necessitar que o algoritmo seja submetido a uma nova etapa de treinamento e teste. Com isso é interessante conhecer o tempo que cada algoritmo leva para a realização deste processo em relação aos demais.

Por fim, como pode ainda ser observado na Tabela 10, diferente dos demais, o presente trabalho fez uso de software livre para o desenvolvimento dos sistemas usados para processamento de imagens e mineração de dados.

5. CONCLUSÃO E TRABALHOS FUTUROS

Os fatores de análise do problema dos grãos ardidos estão relacionados, de uma forma geral, à cor e à textura do grão (PINTO, 2005). Nesta dissertação foi verificada que a combinação desses dois fatores, mesmo computacionalmente, produziu os melhores resultados.

O trabalho de Nil et al. (1997) para detecção de grãos inteiros e grãos quebrados fez uso de sensores óticos e luz estroboscópica para a detecção de objetos em movimento. No entanto, para um processo de classificação de grãos é necessário avaliar a viabilidade de se fazer a coleta e análise individual de grãos. Isso porque uma pequena amostra de grãos coletada para a classificação contém milhares de grãos, o que pode deixar inviável o processo de coleta de imagem do grão um a um. Diferente do trabalho de Nil et al. (1997), a proposta do presente trabalho foi amostrá-las e coletá-las em maior quantidade, em um único processo, com o uso do scanner, como é feito em situações reais. O processo de aquisição apresentado mostrou-se eficiente e viável, podendo ser facilmente expandido para abranger uma coleta usual de classificação.

Assim como nos trabalhos de Steenhoek et al. (2001) e Draganova et al. (2010a), adotou-se, como parâmetros para a realização desta pesquisa, as normas do país em que foi realizado o presente trabalho, ou seja, do Ministério da Agricultura . Essa abordagem permite que os resultados possam ser comparados com aqueles provenientes da legislação vigente, contribuindo para o setor. Ressalva-se que os resultados para a classificação correta superiores a 99% foram obtidos com variados algoritmos.

Embora os trabalhos citados tenham buscado a aplicação de algoritmos específicos como PCA, PNN e Lógica Fuzzy (Steenhoek et al., 2001; Draganova et al., 2010a; Draganova et al., 2010b; Potter et al., 2012), neste trabalho buscou-se identificar quais algoritmos apresentaram maior potencial para serem utilizados para resolução do problema, dada suas características. Acredita-se que a principal contribuição com essa abordagem é justamente em poder servir de referência para futuras pesquisas, por apresentar a performance dos vários algoritmos com este tipo de dados.

Neste trabalho se utilizou dos métodos de Divisão de Amostras e de Validação Cruzada, que são os métodos comumente utilizados em mineração de

dados, e os resultados alcançados possibilitam uma comparação entre os mesmos, servindo também de parâmetro para futuros trabalhos.

Diferente dos trabalhos de Steenhoek et al. (2001), Draganova et al. (2010a), Draganova et al. (2010b) e Potter et al. (2012), buscou-se ainda dar uma abrangência maior na pesquisa, ao se trabalhar com uma quantidade maior de amostras de grãos, combinar e comparar os espaços de cores RGB e HSV (canal H) em conjunto com LBP, testar diferentes resoluções de imagens além de comparar diferentes algoritmos e métodos em mineração de dados.

Com este trabalho foi possível chegar a um modelo matemático que pode ser implementado para a resolução do problema de detecção e classificação de grãos ardidos em milho. A descoberta de um modelo matemático para a detecção e classificação dos grãos é de grande importância já que isto indica que o mesmo pode ser implementado com o uso de qualquer linguagem de programação e embarcado em um sistema de automação computacional.

5.1. TRABALHOS FUTUROS

Como sugestão de trabalho futuro é o desenvolvimento de uma aplicação que reúna todas as funcionalidades apresentadas neste trabalho para automação de classificação de grãos de milho.

Embora este trabalho tenha se concentrado na identificação e classificação de grãos, em trabalhos futuros outros itens podem ser adicionados, como a identificação de matéria estranha, folhas, pedras e insetos nas amostras.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVARENGA, B. S.; et al. Avaliação de técnicas de processamento digital de imagens para a estimativa de áreas de arroz irrigado: um estudo de caso no município de Santa Vitória do Palmar - RS. **In: Simpósio Brasileiro De Sensoriamento Remoto**, 12., 2005, Goiânia. Anais. 2005, p. 3.961-3.966.

ARLOT, S. et al. A survey of cross-validation procedures for model selection. **Statistics surveys**, v. 4, p. 40-79, 2010.

BDS 529-84. **Cereal grains**, 1985.

BRASIL. Ministério da Agricultura, Pecuária e Abastecimento. Secretaria de Defesa Agropecuária. **Manual de Análise Sanitária de Sementes.** Brasília: Mapa/ACS, 2009. 200 p.

BRASIL. Ministério da Agricultura. **Milho.** Acesso em 18 set. 2013.

BRAHNAM, S. et al. Local binary patterns: new variants and applications. **Springer**, 2014.

BREIMAN, L. Random Forests. **Machine Learning**, 45, 5-32, 2001.

BROSNAN, T.; SUN, D. Improving quality inspection of food products by computer vision–a review. **Journal of Food Engineering**, v. 61, n. 1, p. 3-16, 2004.

BURGER, W.; BURGE, M. J. Digital Image Processing: An Algorithmic Approach Using Java. **Springer**, ISBN, v. 184628, p. 3795, 2007.

CAMILO, C. O.; SILVA, J. C. **Mineração de Dados: Conceitos, tarefas, métodos e ferramentas.** Universidade Federal de Goiás (UFC), p. 1-29, 2009.

CARNEIRO, H. S. Comida e sociedade: significados sociais na história da alimentação. **História Questões & Debates**, v. 42, 2005.

CGIAR. **Research Program MAIZE 2014 Annual Report.** Mexico, D.F.: CIMMYT, 2014.

CHENCI, G. P. et al. Uma Introdução á Lógica Fuzzy. **Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica**, v. 1, n. 1, 2011.

COCARO, H.; JESUS, J. C. S. A Agroinformática em Empresas Rurais: algumas tendências. **In: 46th Congress**, July 20-23, 2008, Rio Branco, Acre, Brasil. Sociedade Brasileira de Economia, Administracao e Sociologia Rural (SOBER), 2008.

COLEMAN, T. F.; VAN LOAN, C. **Handbook for matrix computations.** Siam, 1988.

CONAB. **Novo padrão de classificação de milho.** Disponivel em: <http://www.conab.gov.br/OlalaCMS/uploads/arquivos/13_09_02_17_38_42_comu_

194.pdf>. Acesso em Setembro de 2013.

CORSO C.L.; GIBELLINI F. Aplicación de redes bayesianas usando Weka. In: XVII Congreso argentino de ciencias de la computación, La Plata, 2011. **Anais.** La Plata: Universidad Nacional de La Plata, 2011, pag. 939-984.

COSTA, E.; SIMÕES, A. **Inteligência artificial: fundamentos e aplicações.** FCA - Editora, 2008.

COSTA F.O.; et al. Uma abordagem baseada em Redes Neurais Artificiais para o auxílio ao diagnóstico de doenças meningocócicas. **Revista Brasileira de Computação Aplicada**, 2010, v.2, n.1.

COSTA, J. F. A. **Um ambiente gráfico para facilitar tarefas de data mining via ferramenta R**. Dissertação de mestrado em Tecnologias e Sistemas de Informação (área de especialização em Engenharia e Gestão de Sistemas de Informação). Universidade do Minho, Portugal, 2011.

CRUZ, J. C.; et al. **Produção de milho na agricultura familiar.** Sete Lagoas: Embrapa Milho e Sorgo, 2011.

DAYTON, C. M. **Logist Regression Analysis.** Maryland: University of Marylan, 1992.

DE QUEIROZ, J. E. R.; GOMES, H. M. Introdução ao Processamento Digital de Imagens. **RITA**, v. 13, n. 2, p. 11-42, 2006.

DE LIMA, J. R. et al. Comparação de histogramas de imagens digitais para determinação de similaridade em sementes de milho. **Revista de Engenharia e Tecnologia**, v. 4, n. 2, p. Páginas 106-112, 2012.

DELFINO, A. et al. Caracterização da escala de maturação de bananas utilizando técnicas de processamento e análise de imagens digitais. **In: VI Congresso Nacional De Engenharia Mecânica**, 2010.

DEMAAGD, K. et al. **Practical Computer Vision with SimpleCV: The Simple Way to Make Technology See.** O'Reilly Media, Inc., 2012.

DE OLIVEIRA, L. F. et al. Segmentação de Imagens com Fundo Azul utilizando a multiplicaç ao dos Canais HSV. **In: VI Workshop de Visão Computacional**, 2010.

DO AMARAL, V.; THOMAZ, C. E. **Extração e Comparação de Características Locais e Globais para o Reconhecimento Automático de Imagens de Faces.** 2011. Tese de Doutorado. Dissertação de Mestrado, Centro Universitário da FEI, SP, Brasil.

DRAGANOVA, T. et al. Model of Software System for automatic corn kernels Fusarium (spp.) disease diagnostics. **In: Proceedings of the 14th WSEAS international conference on Computers: part of the 14th WSEAS CSCC multiconference-Volume I**. World Scientific and Engineering Academy and Society

(WSEAS), 2010. p. 362-367.

DRAGANOVA, T. et al. An approach for identifying of Fusarium infected maize grains by spectral analysis in the visible and near infrared region, SIMCA models, parametric and neural classifiers. **Int. J. Bioautomat**, v. 14, p. 119-128, 2010.

DUBEY, S. R.; JALAL, A. S. Detection and classification of apple fruit diseases using complete local binary patterns. **In: Computer and Communication Technology (ICCCT)**, 2012 Third International Conference on. IEEE, 2012. p. 346-351.

DZIEKANIAK G. Tecnologias de descoberta de conhecimento na gestão do conhecimento: contextualizações com a sociedade do conhecimento. **Datagrama Zero**, 2010, v. 11, n. 1.

FALCÃO, A. X.; LEITE, N. J. **Fundamentos de processamento de imagem digital**. Tópicos em, 2006. Disponível em: <http://www.ic.unicamp.br/~cpg/material-didatico/mo815/9802/curso/curso.html>. Acesso em Julho de 2014.

FAYYAD, U. et al. From data mining to knowledge discovery in databases. **AI magazine**, v. 17, n. 3, p. 37, 1996.

GONZALEZ, R. C.; WOODS, R. E. **Digital image processing.**3 ed. Reading, Massachusetts: Addison-Wesley Publishing Company, 2008.

HALL, M. et al. The WEKA Data Mining Software: An Update. **SIGKDD Explorations**, 2009, v. 11, n. 1, p. 10-18.

HALLIDAY, D. **Fundamentos de Física 4: óptica e física moderna.** Halliday, Resnick, Walker. 8a edição. LTC, 2009.

HAN, J.; KAMBER, M. **Data Mining - Concepts and Techniques.** The Morgan Kaufmann, 2006.

JAHNE, B. **Digital Image Processing.** Springer, 2005.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. **In Proc. of the 11th UAI**, pages 338–345, Montreal, Canada, 1995.

KASZNAR, I. K.; GOLÇAVES, B. M. L. **Regressão múltipla: uma digressão sobre seus usos.** 2011.

KANADE, T; JAIN, A.; RATHA, N. K. **Audio- and Video-Based Biometric Person Authentication**: 5th International Conference, AVBPA 2005, Hilton Rye Town, NY, USA, July 20-22, 2005, Proceedings. Springer Science & Business Media. 2005.

KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. **In: Ijcai.** 1995. p. 1137-1145.

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. **Biometrics**, p. 159-174, 1977.

Liu, J.; Paulsen, M. R. Corn whiteness measurement and classification using machine vision. **In 1997 ASAE Annual International Meeting**, Paper No. 973045. St. Joseph, Michigan, USA: ASAE, 1997.

LORENA, A. C.; DE CARVALHO, A. C. P. L. F. Uma introdução às support vector machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43-67, 2007.

MAIMON, O. Z.; ROKACH, L. **Data mining and knowledge discovery handbook.** New York: Springer, 2010.

MACHADO, R. T. M. Tecnologia da informação e competitividade em sistemas agroindustriais: um estudo exploratório. **Revista Brasileira de Agroinformática**, v. 1, n. 1, p. 66-76, 1998.

MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à visão computacional usando opencv. **Revista de Informática Teórica e Aplicada**, v. 16, n. 1, p. 125-160, 2009.

MARQUES FILHO, O.; VIEIRA NETO, H. **Processamento digital de imagens.** Brasport, 1999.

MAZOYER, M.; ROUDART, L. **História das agriculturas no mundo. Do neolítico à crise contemporânea.** São Paulo: Editora UNESP, 2008.

MEIRA, C. A. A. et al. Agroinformática: qualidade e produtividade na agricultura. **Cadernos de Ciência & Tecnologia**, v. 13, n. 2, p. 175-194, 1996.

MENDES, C. A. B.; VEGA, F. A. C. Técnicas de Regressão Logística Aplicada à análise ambiental. **GEOGRAFIA (Londrina)**, v. 20, n. 1, p. 5-30, 2011.

MENDES, M. et al. Comportamento de híbridos de milho inoculados com os fungos causadores do complexo grãos ardidos e associação com parâmetros químicos e bioquímicos. **Ambiência**, Guarapuava, 2012, v. 8, n. 2, p. 275-292.

MERY, D. et al. Automated fish bone detection using X-ray imaging. **Journal of Food Engineering**, v. 105, n. 3, p. 485-492, 2011.

NI, B. et al. Design of an automated corn kernel inspection system for machine vision. **Transactions of the ASAE**, v. 40, n. 2, p. 491-497, 1997.

NG, H. F. et al. Machine vision evaluation of corn kernel mechanical and mould damage. **In 1997 ASAE Annual International Meeting**, Paper No. 973047. S t. Joseph, Michigan, USA: ASAE, 1997.

OHTA, D. L. A. et al. Um protótipo de uma ferramenta de detecção automática de alterações posturais baseada em imagens. **In: WVC'2006 - II Workshop de Visão Computacional**, São Carlos, SP, 2006, p. 370-373.

OJALA, T. et al. A comparative study of texture measures with classification based

on featured distributions. **Pattern recognition**, v. 29, n. 1, p. 51-59, 1996.

PAES, M. C. D. Aspectos Físicos, Químicos e Tecnológicos do Grão de Milho. **Circular Técnica, 75-EMBRAP A. Ministério da Agricultura, Pecuária e Abastecimento.** 2006.

PIETIKAINEN, M. Local binary patterns. **Scholarpedia**, v. 5, n. 3, p. 9775, 2010.

PIMENTA, A. et al. WEKA-G: mineração de dados paralela em grades computacionais. **Revista de Sistemas de Informação da FSMA**, v. 4, p. 2, 2009.

PINTO, N. F. J. A. **Qualidade sanitária de grãos de milho.** Sete Lagoas: Embrapa Milho e Sorgo, 2001.

PONTIL, M.; VERRI, A. Properties of Support Vector Machines. **Neural Computation**, 10, 955-974, 1998.

POTTER, P. et al. **Automatic Visual Inspection of Corn Kernels Using Principal Component Analysis.** 2012.

POURREZA, A. et al. Identification of nine Iranian wheat seed varieties by textural analysis with image processing. **Computers and Electronics in Agriculture**, v. 83, p. 102-108, 2012.

PYTHON SOFTWARE FOUNDATION. **The python programming language.** 2012. Available at <http://www.python.org>

QUILICI-GONZALEZ, J. A.; DE ASSIS ZAMPIROLLI, F. **Sistemas Inteligentes e Mineração de Dados.** Triunfal Gráfica e Editora, 1.a Edição, 148p, 2014.

RINGNÉR M. What is Principal Component Analysis ? **Nature Biotechnology**, 26(3), 303-304, 2008.

RIBEIRO, S. S. et al. Método computacional para quantificação de coloração vermelha presente no fruto do morango. **In: II Simpósio Paranaense de Fruticultura**, 2014, Ponta Grossa. Anais - II Simpósio Paranaense de Fruticultura, 2014. p. 131-131.

ROCHA, M. et al. Evolution of neural networks for classification and regression. **Neurocomputing**, v. 70, n. 16, p. 2809-2816, 2007.

SCOLARI, D. D. G. Produção agrícola mundial: o potencial do Brasil. **Revista da Fundação Milton Campos**, Brasília, DF, n. 25, p. 09-86, 2006.

SATO, L. Y.; et al. Análise comparativa de algoritmos de árvore de decisão do sistema WEKA para classificação do uso e cobertura da terra. **In: XVI Simpósio Brasileiro de Sensoriamento Remoto**, Foz do Iguaçu, 2013. Anais. Foz do Iguaçu: INPE.

SHAPIRO, L.; STOCKMAN, G. C. **Computer Vision.** ed: Prentice Hall, 2001.

SOLEM, J. E. **Programming Computer Vision with Python: Tools and algorithms for analyzing images. "** O'Reilly Media, Inc.", 2012.

SOUZA, P. B. **Uma estratégia baseada em algoritmos de mineração de dados para validar plano de operação de voo a partir de predições de estados dos satélites do INPE.** São José dos Campos, SP, Brasil: INPE, 2011.

STEENHOEK, L. W. Probabilistic neural networks for segmentation of features in corn kernel images. **Applied engineering in agriculture**, v. 17, n. 2, p. 225-234, 2001.

STEENHOEK, L. W. et al. Implementing a computer vision system for corn kernel damage evaluation. **Applied Engineering in Agriculture**, v. 17, n. 23, p. 235, 2001.

SURAL, S. et al. Segmentation and histogram generation using the HSV color space for image retrieval. **In: Image Processing**. 2002. Proceedings. 2002 International Conference on. IEEE, 2002. p. II-589-II-592 vol. 2.

TAKATSUI, F. **Sistema CIE Lab: análise computacional de fotografias.** 2011.

TAVARES, L. G. et al. Estudo comparativo de métodos de aprendizado de máquina na detecção de regiões promotoras de genes de escherichia coli. **Anais do I Simpósio Brasileiro de Inteligência Computacional**, p. 8-11, 2007.

TURBAN, E. et al. **Decision support and business intelligence systems.** Pearson Education India, 2010.

VIANA, G. Milho: novos sistemas de produção e busca por maiores produtividades provocam aumento da severidade das doenças. **Jornal Eletrônico da Embrapa Milho e Sorgo.** Ano 03 - Edição 19 - Novembro de 2009. Sete Lagoas-MG.

WARD SYSTEMS GROUP. **Neuro Windows Neural Network Dynamic Link Library.** Frederick, Md.: Ward Systems Group, –Inc. 1997.

VAN WATERBEEMD, H. Discriminant analysis for activity prediction. **Method and principles in medicinal chemistry**, v. 2, p. 265-282, 1995.

WEBER, Leo; KLEIN, Pedro Antonio Trierweiler. **Aplicação da lógica fuzzy em software e hardware.** Editora da ULBRA, 2003.

WIGGERS, K. L.; et al. Análise de abordagens para extração de características de sementes usando processamento digital de imagens. **In: IX Congresso Brasileiro De Agroinformática**, 2013, Cuiabá - MT. Anais. 2013.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical Machine Learning Tools and Techniques.** Morgan Kaufmann, San Francisco, CA, 2th edition, 2005.

APÊNDICE A – Programa SEGMENTADOR

No Algoritmo 1 é apresentado o código do programa SEGMENTADOR que foi utilizado para o processo de leitura e segmentação das imagens geradas pela aquisição realizada com o scanner.

ALGORITMO 1 – Código do programa SEGMENTADOR.

```
from SimpleCV import Image,Color, Display, np
import glob, math, os, sys, time
import cv
import numpy as np

class Img:
    files    = []
    dpi      = '75'
    origem   = ''
    destino  = ''
    color    = Color.PUCE
    detected = False

    def generation(self):
        w       = 150
        h       = 75
        fmin    = 250
        fmax    = 1000
        vmin    = 250
        vmax    = 1000

        if(self.dpi=='300'):
            w       = 300
            h       = 150
            fmin    = 1000
            fmax    = 30000
            vmin    = 1000
            vmax    = 30000

        if(self.dpi=='600'):
            w       = 600
            h       = 400
            fmin    = 1000
            fmax    = 130000
            vmin    = 1000
            vmax    = 130000


        for file in self.files:
            frente      = file+'-F.png'
            verso       = file+'-V.png'
            img_frente  = Image(self.origem+frente)
            img_verso   = Image(self.origem+verso)
            img_frente  = img_frente.crop(5,5,img_frente.width-10,img_frente.height-5).medianFilter()
            img_verso   = img_verso.crop(5,5,img_frente.width-10,img_frente.height-5).medianFilter()
            mask_frente = img_frente.colorDistance(self.color).medianFilter().binarize().erode(1).invert()
            mask_verso  = img_verso.colorDistance(self.color).medianFilter().binarize().erode(1).invert()
            seed_frente = img_frente - mask_frente.invert()
            seed_verso  = img_verso  - mask_verso.invert()

            color_dist_frente = seed_frente.colorDistance(self.color).invert()
            color_dist_verso  = seed_verso.colorDistance(self.color).invert()

            blobs_frente = seed_frente.findBlobs(minsize=fmin,maxsize=fmax)
            blobs_verso  = seed_verso.findBlobs(minsize=vmin,maxsize=vmax)

            if self.detected:
                blobs_frente.draw(color=Color.GREEN, width=2)
                blobs_verso.draw(color=Color.GREEN, width=2)
                seed_frente.addDrawingLayer(color_dist_frente.dl())
                seed_verso.addDrawingLayer(color_dist_verso.dl())

                seed_frente.save(self.origem+file+'-F-DETECTED.png')
                seed_verso.save(self.origem+file+'-V-DETECTED.png')

            print len(blobs_frente),len(blobs_verso)

            if len(blobs_frente) < len(blobs_verso):
                lim = len(blobs_frente)
            else:
                lim = len(blobs_verso)

            n_frente = 0
            n_verso  = 0

            for n in range(0,lim):
                ncorte = 87
                if "SADIOS-T-IMG08" in file:
                    ncorte=63

                img_seed_f = seed_frente.crop(blobs_frente[n_frente])
                img_seed_v = seed_verso.crop(blobs_verso[n_verso])

                if self.dpi == '75':
                    if img_seed_f.height < 20 or img_seed_f.height > 50:
                        n_frente+=1
                    if img_seed_v.height < 20 or img_seed_v.height > 50:
                        n_verso+=1

```

```
91                 if self.dpi == '300':
92                     if img_seed_f.height < 20 or img_seed_f.height > 200:
93                         n_frente+=1
94                     if img_seed_v.height < 20 or img_seed_v.height > 200:
95                         n_verso+=1
96                     if img_seed_f.width < 20 or img_seed_f.width > 200:
97                         n_frente+=1
98                     if img_seed_v.width < 20 or img_seed_v.width > 200:
99                         n_verso+=1
100
101                                             print   n,n_frente,n_verso,"F:   ",img_seed_f.height,img_seed_f.width,"
102 V:",img_seed_v.height,img_seed_v.width
103
104                 if(n<=ncorte):
105                     nwfile = file+'-'+str(n+1).zfill(2)+'.png'
106                     img_seed_f = seed_frente.crop(blobs_frente[n_frente])
107                     img_seed_v = seed_verso.crop(blobs_verso[n_verso])
108                     nw_image  = Image((w,h))
109
110                     x1 = ((w/2)-img_seed_f.width)/2
111                     y1 = (h-img_seed_f.height)/2
112                     nw_image.dl().blit(img_seed_f, (x1,y1))
113
114                     x2 = (((w/2)-img_seed_v.width)/2)+(w/2)
115                     y2 = (h-img_seed_v.height)/2
116                     nw_image.dl().blit(img_seed_v, (x2,y2))
117
118                     nw_image.save(self.destino+nwfile)
119                 else:
120                     print "-->",blobs_frente[n_frente].height,blobs_verso[n_verso].height
121
122                 n_frente+=1
123                 n_verso+=1
124
125 def geraImg():
126
127     nwImg = Img()
128
129     nwImg.files    = ["ARDIDOS-A-IMG01","ARDIDOS-C-IMG01","ARDIDOS-C-IMG02","ARDIDOS-C-IMG03"]
130     nwImg.files   += ["ARDIDOS-T-IMG01","ARDIDOS-T-IMG02","ARDIDOS-T-IMG03","ARDIDOS-T-IMG04"]
131     nwImg.files   += ["ARDIDOS-T-IMG05","ARDIDOS-T-IMG06","ARDIDOS-T-IMG07","ARDIDOS-T-IMG08"]
132     nwImg.files   += ["SADIOS-A-IMG01","SADIOS-C-IMG01","SADIOS-C-IMG02","SADIOS-C-IMG03"]
133     nwImg.files   += ["SADIOS-T-IMG01","SADIOS-T-IMG02","SADIOS-T-IMG03","SADIOS-T-IMG04"]
134     nwImg.files   += ["SADIOS-T-IMG05","SADIOS-T-IMG06","SADIOS-T-IMG07","SADIOS-T-IMG08"]
135     nwImg.dpi      = '75'
136     nwImg.origem   = '/home/sergio/Pictures/aquisicoes/resultado/'+nwImg.dpi+'DPI/'
137     nwImg.destino  = nwImg.origem+'IMG/'
138     nwImg.color    = Color.BLUE # Background color
139
140     nwImg.detected = False
141
142     start_time = time.time()
143     nwImg.generation()
144     print "--- elapsed time: ", (time.time() - start_time)/60 , " minutes ---"
145
146
147 if __name__ == "__main__" :
148     geraImg()
149
150
```

As linhas iniciais do programa, 1 a 5, são responsáveis pelo carregamento das bibliotecas e classes do Python para ativar os recursos de processamento digital de imagens. Entre as linhas 6 e 124 tem-se o código da classe Img que possui seis atributos e um método. O método generation() da classe é quem faz todo o processo de segmentação com base nas informações dos atributos. Nas linhas 15 a 36 são definidas alguns valores que serão aplicados à geração das imagens, com base nas informações de dpi fornecidas.

O laço, a partir da linha 39, é responsável por fazer a leitura dos arquivos de imagens com os grãos, e identificar as imagens correspondentes de frente e verso. Para o processo de segmentação primeiro é aplicado um filtro de mediana, linhas 44 e 45. A separação dos objetos de interesse do fundo é feito um cálculo de distância, tendo como base a cor azul do fundo, linhas 46 e 47. Em seguida é aplicado

novamente um filtro de mediana, para obter a binarização, seguida de uma erosão para redução de ruidos na imagem. Com a inversão de cor feita, nas linhas 48 e 49 os grãos são detectados.

A partir da linha 57 os grãos detectados são unidos, frente e verso, para formar as imagens individuais de cada grão, que serão salvas no caminho previamente informado.

Entre as linhas 125 e 144 tem-se uma função principal denominada geraImg(), que é carregada assim que o programa é executado, conforme a linha 147. Na função geraImg() é feito o instanciamento da classe Img, são colocados valores aos seus atributos, e é executado o método generation() da classe.

APÊNDICE B – Programa EXTRATOR

No Algoritmo 2 está o código do programa EXTRATOR que foi desenvolvido para realizar o processo de extração de características das imagens e gerar a base de dados para o processo de mineração de dados.

ALGORITMO 2 – Código do programa EXTRATOR.

```
from SimpleCV import Image,Color, Display, np
import glob, math, os, sys, time
import cv
import numpy as np

def conv_lbp(imgGRAY):
    arrImg = imgGRAY.getNumpy()
    arrImg =  (1<<7) * (arrImg[0:-2,0:-2] >= arrImg[1:-1,1:-1]) \
            + (1<<6) * (arrImg[0:-2,1:-1] >= arrImg[1:-1,1:-1]) \
            + (1<<5) * (arrImg[0:-2,2:] >= arrImg[1:-1,1:-1]) \
            + (1<<4) * (arrImg[1:-1,2:] >= arrImg[1:-1,1:-1]) \
            + (1<<3) * (arrImg[2:,2:] >= arrImg[1:-1,1:-1]) \
            + (1<<2) * (arrImg[2:,1:-1] >= arrImg[1:-1,1:-1]) \
            + (1<<1) * (arrImg[2:,:-2] >= arrImg[1:-1,1:-1]) \
            + (1<<0) * (arrImg[1:-1,:-2] >= arrImg[1:-1,1:-1])
    return np.bincount(arrImg.ravel())

def percentile(h,p):
     s = h.sum()
     k = ((s-1) * p/100.)+1
     dw = np.floor(k)
     up = np.ceil(k)
     hc = np.cumsum(h)
     if isinstance(p, int):
        k1 = np.argmax(hc>=dw)
        k2 = np.argmax(hc>=up)
     else:
        k1 = np.argmax(hc>=dw[:,np.newaxis],axis=1)
        k2 = np.argmax(hc>=up[:,np.newaxis],axis=1)
     d0 = k1 * (up-k)
     d1 = k2 * (k -dw)
     return np.where(dw==up,k1,d0+d1)

def hstats(h):
    hn = 1.0*h/h.sum() # compute the normalized image histogram
    v = np.zeros(11) # number of statistics
    # compute statistics
    n = len(h) # number of gray values
    v[0]  = np.sum((np.arange(n)*hn)) # mean/media
    v[1]  = np.sum(np.power((np.arange(n)-v[0]),2)*hn) # variance/variancia
    v[2]  = np.sum(np.power((np.arange(n)-v[0]),3)*hn)/(np.power(v[1],1.5))# skewness/obliquidade
    v[3]  = np.sum(np.power((np.arange(n)-v[0]),4)*hn)/(np.power(v[1],2))-3# kurtosis/curtose
    v[4]  = -(hn[hn>0]*np.log(hn[hn>0])).sum() # entropy/entropia
    v[5]  = np.argmax(h) # mode/moda
    v[6:]  = percentile(h,np.array([1,10,50,90,99])) # 1,10,50,90,99% percentile/percentil
    data  = str(v[0])
    data += ","+str(v[1])
    data += ","+str(v[2])
    data += ","+str(v[3])
    data += ","+str(v[4])
    data += ","+str(v[5])
    data += ","+str(v[6])
    return data

class Arff:
    relation   = ''
    attributes = []
    datas      = []

    def __init__(self):
        self.relation   = ''
        self.attributes = []
        self.datas      = []

    def addAttribute( self,attribute ):
            self.attributes.append(attribute)

    def addData( self,data ):
            self.datas.append(data)

    def generation( self,file ):
            f2w = open(file,"w")
            f2w.write( "@relation "+self.relation+"\n" )

            for attribute in self.attributes:
                f2w.write( "@attribute "+attribute+"\n" )

            f2w.write( "@data "+"\n" )
            for data in self.datas:
                f2w.write( data+"\n" )

            f2w.close()

def getfromhist(histC):
    total = 0
    data  = ""
    for color in histC:
        total += color
        if(data==""):
```

```
            data += str(color)+".0"
        else:
            data += ","+str(color)+".0"

    #data += ","+str(total)+".0"

    return data


def
geraArff( prefixo="COLOUR",dpi="75",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=False,gHISTLBP=False,gSTAT=False
,filter="" ):

    objArff = Arff()

    if gHISTR:
        prefixo += "-R"
    if gHISTG:
        prefixo += "-G"
    if gHISTB:
        prefixo += "-B"
    if gHISTHSV:
        prefixo += "-HSV"
    if gHISTLBP:
        prefixo += "-LBP"
    if gSTAT:
        prefixo += "-STAT"

    prefixo += "-"+dpi

    objArff.relation = prefixo

    total    = 1000
    dir      = '/home/sergio/Pictures/aquisicoes/resultado/'+dpi+'DPI/IMG/'
    texture  = range(0,255)
    tipos    = ["SADIO","ARDIDO"]
    maskimg  = "*.png"

    if gHISTR:
        colors = range(0,256)
        for n in colors:
            objArff.addAttribute( "colorR_"+str(n)+" real" )

        if gSTAT:
            objArff.addAttribute( "Rmedia real" )
            objArff.addAttribute( "Rvariancia real" )
            objArff.addAttribute( "Robliquidade real" )
            objArff.addAttribute( "Rcurtose real" )
            objArff.addAttribute( "Rentropia real" )
            objArff.addAttribute( "Rmoda real" )
            objArff.addAttribute( "Rpercentil real" )

    if gHISTG:
        colors = range(0,256)
        for n in colors:
            objArff.addAttribute( "colorG_"+str(n)+" real" )

        if gSTAT:
            objArff.addAttribute( "Gmedia real" )
            objArff.addAttribute( "Gvariancia real" )
            objArff.addAttribute( "Gobliquidade real" )
            objArff.addAttribute( "Gcurtose real" )
            objArff.addAttribute( "Gentropia real" )
            objArff.addAttribute( "Gmoda real" )
            objArff.addAttribute( "Gpercentil real" )

    if gHISTB:
        colors = range(0,256)
        for n in colors:
            objArff.addAttribute( "colorB_"+str(n)+" real" )

        if gSTAT:
            objArff.addAttribute( "Bmedia real" )
            objArff.addAttribute( "Bvariancia real" )
            objArff.addAttribute( "Bobliquidade real" )
            objArff.addAttribute( "Bcurtose real" )
            objArff.addAttribute( "Bentropia real" )
            objArff.addAttribute( "Bmoda real" )
            objArff.addAttribute( "Bpercentil real" )

    if gHISTHSV:
        colors = range(0,256)
        for n in colors:
            objArff.addAttribute( "colorHSV_"+str(n)+" real" )

        if gSTAT:
            objArff.addAttribute( "HSVmedia real" )
            objArff.addAttribute( "HSVvariancia real" )
            objArff.addAttribute( "HSVobliquidade real" )
            objArff.addAttribute( "HSVcurtose real" )
            objArff.addAttribute( "HSVentropia real" )
            objArff.addAttribute( "HSVmoda real" )
            objArff.addAttribute( "HSVpercentil real" )

    if gHISTLBP:
        colors = range(0,256)
        for n in colors:
            objArff.addAttribute( "colorLBP_"+str(n)+" real" )

        if gSTAT:
            objArff.addAttribute( "LBPmedia real" )
```

```
192             objArff.addAttribute( "LBPvariancia real" )
193             objArff.addAttribute( "LBPobliquidade real" )
194             objArff.addAttribute( "LBPcurtose real" )
195             objArff.addAttribute( "LBPentropia real" )
196             objArff.addAttribute( "LBPmoda real" )
197             objArff.addAttribute( "LBPpercentil real" )
198
199     objArff.addAttribute( "classe { SADIO, ARDIDO}" )
200
201     for tipo in tipos:
202         files = glob.glob( dir+"/"+tipo+maskimg)
203         ardidos = 1
204         sadios  = 1
205         for file in files:
206             fileName = file.split("/",8)[8]
207             if (("ARDIDOS" in file and ardidos<=total) or ("SADIOS" in file and sadios<=total))
208                   and (filter=="" or (fileName in filter)):
209                 if "ARDIDOS" in file:
210                     ardidos+=1
211                 else:
212                     sadios+=1
213
214                 data = ""
215                 img  = Image(file)
216
217                 if gHISTR:
218                     ( hR, hG, hB ) = img.splitChannels(False)
219                     histC = np.array(hR.histogram(256))
220                     if data<>"":
221                         data +=","
222                     data += getfromhist(histC)
223                     if gSTAT:
224                         data += ","+hstats(histC)
225
226                 if gHISTG:
227                     ( hR, hG, hB ) = img.splitChannels(False)
228                     histC = np.array(hG.histogram(256))
229                     if data<>"":
230                         data +=","
231                     data += getfromhist(histC)
232                     if gSTAT:
233                         data += ","+hstats(histC)
234
235                 if gHISTB:
236                     ( hR, hG, hB ) = img.splitChannels(False)
237                     histC = np.array(hB.histogram(256))
238                     if data<>"":
239                         data +=","
240                     data += getfromhist(histC)
241                     if gSTAT:
242                         data += ","+hstats(histC)
243
244                 if gHISTHSV:
245                     histC = img.hueHistogram(256)
246                     if data<>"":
247                         data +=","
248                     data += getfromhist(histC)
249                     if gSTAT:
250                         data += ","+hstats(histC)
251
252                 if gHISTLBP:
253                     histLBP = conv_lbp(img)
254                     if data<>"":
255                         data +=","
256                     data += getfromhist(histLBP)
257                     if gSTAT:
258                         data += ","+hstats(histLBP)
259
260                 data += ","+tipo
261                 objArff.addData(data)
262
263     objArff.generation("/home/sergio/base-"+prefixo+"dpi.arff")
264
265     del objArff,data,gHISTR,gHISTG,gHISTB,gHISTHSV,gHISTLBP,gSTAT
266
267
268 if __name__ == "__main__" :
269     start_time = time.time()
270     data = ""
271     with open ("/home/sergio/Pictures/aquisicoes/resultado/1000ToModel.txt", "r") as myfile:
272         data=myfile.read().replace('\n', ' ')
273
274     print "75-RGB"
275     geraArff(dpi="75",gHISTR=True,gHISTG=True,gHISTB=True,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
276     print "300-RGB"
277     geraArff(dpi="300",gHISTR=True,gHISTG=True,gHISTB=True,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
278     print "600-RGB"
279     geraArff(dpi="600",gHISTR=True,gHISTG=True,gHISTB=True,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
280
281     print "75-R"
282     geraArff(dpi="75",gHISTR=True,gHISTG=False,gHISTB=False,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
283     print "300-R"
284     geraArff(dpi="300",gHISTR=True,gHISTG=False,gHISTB=False,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
285     print "600-R"
286     geraArff(dpi="600",gHISTR=True,gHISTG=False,gHISTB=False,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
287
288     print "75-G"
289     geraArff(dpi="75",gHISTR=False,gHISTG=True,gHISTB=False,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
290     print "300-G"
291     geraArff(dpi="300",gHISTR=False,gHISTG=True,gHISTB=False,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
292     print "600-G"
```

```
293     geraArff(dpi="600",gHISTR=False,gHISTG=True,gHISTB=False,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
294
295     print "75-B"
296     geraArff(dpi="75",gHISTR=False,gHISTG=False,gHISTB=True,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
297     print "300-B"
298     geraArff(dpi="300",gHISTR=False,gHISTG=False,gHISTB=True,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
299     print "600-B"
300     geraArff(dpi="600",gHISTR=False,gHISTG=False,gHISTB=True,gHISTHSV=False,gHISTLBP=False,gSTAT=False,filter=data)
301
302     print "75-HSV"
303     geraArff(dpi="75",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=True,gHISTLBP=False,gSTAT=False,filter=data)
304     print "300-HSV"
305     geraArff(dpi="300",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=True,gHISTLBP=False,gSTAT=False,filter=data)
306     print "600-HSV"
307     geraArff(dpi="600",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=True,gHISTLBP=False,gSTAT=False,filter=data)
308
309     print "75-LBP"
310     geraArff(dpi="75",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=False,gHISTLBP=True,gSTAT=False,filter=data)
311     print "300-LBP"
312     geraArff(dpi="300",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=False,gHISTLBP=True,gSTAT=False,filter=data)
313     print "600-LBP"
314     geraArff(dpi="600",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=False,gHISTLBP=True,gSTAT=False,filter=data)
315
316     print "75-HSVxLBP"
317     geraArff(dpi="75",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=True,gHISTLBP=True,gSTAT=False,filter=data)
318     print "300-HSVxLBP"
319     geraArff(dpi="300",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=True,gHISTLBP=True,gSTAT=False,filter=data)
320     print "600-HSVxLBP"
321     geraArff(dpi="600",gHISTR=False,gHISTG=False,gHISTB=False,gHISTHSV=True,gHISTLBP=True,gSTAT=False,filter=data)
322
323     print "--- elapsed time: ", (time.time() - start_time)/60 , " minutes ---"
324
325
```

Nas linhas iniciais do programa, linhas 1 a 4, são carregadas as bibliotecas e classes do Python. A partir da linha 6 tem-se a definição da função conv_lbp() que é quem faz o cálculo do padrão LBP, usando como entrada uma imagem em escala de cinza.

Na linha 18 tem-se a função para cálculo de percentil() e na linha 34 a função de cálculo de valores estatísticos, como média, variância, obliquidade, curtose, entropia e moda. Essas funções não foram utilizadas nesse trabalho, porém estão disponíveis para trabalhos futuros.

Entre as linhas 55 e 83 está o código da classe Arff, que faz a montagem e a preparação dos dados para a geração do arquivo para o Weka.

A partir da linha 85 tem-se a função getfromhist(), que gera uma linha contendo o resultado do histograma para cada cor, que é posteriormente repassada para a classe Arff. A função geraArff(), que é a função principal e responsável por instanciar a classe Arff e chamar os métodos de alimentação de dados está a partir da linha 100.

Na linha 263 o arquivo no padrão Weka é por fim gerado e na linha 265 são removidas da memórias as variáveis utilizadas. Esse comando é executado devido à quantidade de informações armazenada nessas variáveis, liberando espaço na memória e melhorando o desempenho do programa.

O comando presente na linha 268 é responsável pela inicialização do programa. Nas linhas 269 a 272 são definidas algumas variáveis de controle e de

dados.

Pode-se observar que os comandos das linhas 274 e 275 se repetem nas linhas subsequentes. A linha 274 apenas imprime uma mensagem informando quais imagens estão sendo processadas, e a linha 275 contém a chamada para a principal função do programa, geraArff(), passando os parâmetros necessários para o processamento da base a ser gerada.

APÊNDICE C – Programa TESTMODEL

Em Algoritmo 3 é apresentado o código do programa TESTMODEL. Esse programa foi responsável por testar o modelo gerado pelo processo de mineração de dados e foi parametrizado para testar a base com base com referência aos parêmtros fornecidos pelo MAPA (CONAB, 2013) para classificação de grãos no Brasil.

ALGORITMO 3 – Código do programa TESTMODEL.

```
from SimpleCV import Image,Color, Display, np
import glob, math, os, sys, time
import cv
import numpy as np

def conv_lbp(imgGRAY):
    arrImg = imgGRAY.getNumpy()
    arrImg =  (1<<7) * (arrImg[0:-2,0:-2] >= arrImg[1:-1,1:-1]) \
            + (1<<6) * (arrImg[0:-2,1:-1] >= arrImg[1:-1,1:-1]) \
            + (1<<5) * (arrImg[0:-2,2:] >= arrImg[1:-1,1:-1]) \
            + (1<<4) * (arrImg[1:-1,2:] >= arrImg[1:-1,1:-1]) \
            + (1<<3) * (arrImg[2:,2:] >= arrImg[1:-1,1:-1]) \
            + (1<<2) * (arrImg[2:,1:-1] >= arrImg[1:-1,1:-1]) \
            + (1<<1) * (arrImg[2:,:-2] >= arrImg[1:-1,1:-1]) \
            + (1<<0) * (arrImg[1:-1,:-2] >= arrImg[1:-1,1:-1])
    return np.bincount(arrImg.ravel())

def classify(hsv,lbp):
    c0  = -1.86
    c0 += float(hsv[205]) * - 0.09
    c0 += float(hsv[217]) * - 0.08
    c0 += float(hsv[220]) * - 0.07
    c0 += float(hsv[230]) * - 0.04
    c0 += float(lbp[50])  * - 0.07
    c0 += float(lbp[54])  * - 0.03
    c0 += float(lbp[66])  *   0.06
    c0 += float(lbp[125]) *   0.03
    c0 += float(lbp[216]) * - 0.03

    c1  = 1.86
    c1 += hsv[205] *   0.09
    c1 += hsv[217] *   0.08
    c1 += hsv[220] *   0.07
    c1 += hsv[230] *   0.04
    c1 += lbp[50]  *   0.07
    c1 += lbp[54]  *   0.03
    c1 += lbp[66]  * - 0.06
    c1 += lbp[125] * - 0.03
    c1 += lbp[216] *   0.03

    return c0,c1

filter = ""
teste = "500-99x1-ToTest.txt"
with open ("/home/sergio/Pictures/aquisicoes/resultado/"+teste, "r") as myfile:
    filter=myfile.read().replace('\n', ' ')

dir = '/home/sergio/Pictures/aquisicoes/resultado/300DPI/IMG/'

files = glob.glob( dir+"/*.png")

s      = 0
a      = 0
ac_s   = 0
ac_a   = 0

for file in files:
    fileName = file.split("/",8)[8]
    if fileName in filter:
        img  = Image(file)
        histC = img.hueHistogram(256)
        histLBP = conv_lbp(img)

        c0,c1 = classify(histC,histLBP)

        if fileName[0]=="S":
            s+=1
        else:
            a+=1

        if c0>=-6.5 and fileName[0]=="S":
            ac_s+=1

        if c1>3.5 and fileName[0]=="A":
            ac_a+=1

print "Total de Sadias => ",s,", Acertou => ",ac_s,", Errou => ",s-ac_s
print "Total de Ardidos => ",a,", Acertou => ",ac_a,", Errou => ",a-ac_a
```

Nas linhas iniciais do programa, 1 a 4, são carregadas as bibliotecas e classes do Python. Entre as linhas 6 e 16 está o código da função conv_lbp(), que gera os dados de textura no padrão LBP, com base na imagem em escala de cinza recebida como parâmetro. As linhas 18 a 41 apresentam o código da função classify() com o modelo extraído do Weka. Essa função recebe como parâmetro os histogramas de HSV e LBP e faz a classificação com base no modelo retornando o resultado da classificação.

Entre as linhas 44 e 58 são informadas as variáveis de parametrização do programa. Na linha 44 é informado o arquivo com as imagens que serão usadas para o teste do modelo. Nas linhas 45 e 46 é feita a leitura dessas imagens, que são armazenadas em uma matriz.

Na linha 57 é iniciado o laço principal, que faz a leitura de todas as imagens a serem classificadas, filtrando-as de acordo com a matriz de filtro definida anteriormente.

As linhas 72 e 73 verificam se a imagem pertence à classe de grãos sadios e incrementa a variável de contagem de grãos sadios e as linhas 75 e 76 faz o mesmo para o caso dos grãos ardidos. As linhas 79 e 80 apresentam o resultado da classificação.