

TOWARDS AN AUTOMATIC CLASSIFICATION OF TAHITI LIME

Ellen Marques Giacometti

Dissertation submitted to the Postgraduate Program in EEL of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of master.

Advisor: Professor D.Sc. Gabriel Matos Araujo

Rio de Janeiro,

Dec 2024

TOWARDS AN AUTOMATIC CLASSIFICATION OF TAHITI LIME

Dissertation submitted to the Postgraduate Program in EEL of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of master.

Ellen Marques Giacometti

Examining jury:

Documento assinado digitalmente
gov.br
GABRIEL MATOS ARAUJO
Data: 28/03/2025 16:35:57-0300
Verifique em https://validar.iti.gov.br

President, Professor D.Sc. Gabriel Matos Araujo (CEFET/RJ) (Advisor)

Documento assinado digitalmente
gov.br
AMARO AZEVEDO DE LIMA
Data: 28/03/2025 17:36:36-0300
Verifique em https://validar.iti.gov.br

Professor Ph.D. Amaro Azevedo de Lima (CEFET/RJ)

Documento assinado digitalmente
gov.br
DIEGO BARRETO HADDAD
Data: 28/03/2025 22:00:40-0300
Verifique em https://validar.iti.gov.br

Professor D.Sc. Diego Barreto Haddad (CEFET/RJ)

Documento assinado digitalmente
gov.br
FELIPE DA ROCHA HENRIQUES
Data: 28/03/2025 22:29:41-0300
Verifique em https://validar.iti.gov.br

Professor D.Sc. Felipe da Rocha Henriques (CEFET/RJ)

Documento assinado digitalmente
gov.br
MURILLO FERREIRA DOS SANTOS
Data: 30/03/2025 12:16:22-0300
Verifique em https://validar.iti.gov.br

Professor P. Eng. Murillo Ferreira dos Santos (CEFET/MG)

Rio de Janeiro,

Dec 2024

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

G429 Giacometti, Ellen Marques
Towards an automatic classification of tahiti lime / Ellen Marques Giacometti. — 2024.
75f. : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2024.
Bibliografia : f. 71-75
Orientador: Gabriel Matos Araujo

1. Visão por computador. 2. Lima acida taiti - Controle de qualidade. 3. Processamento de imagens - Técnicas digitais . 4. Sistema de reconhecimento de padrões. I. Araujo, Gabriel Matos. (Orient.). II. Título.

CDD 006.37

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

DEDICATION

To my parents, for their unconditional love and support.

.

ACKNOWLEDGMENTS

I dedicate this work to those who have supported me throughout this journey, offering their guidance, encouragement, and belief in my potential.

To my parents, whose unwavering love and support have been the foundation of my academic and personal growth. Your sacrifices, patience, and constant encouragement have given me the strength to pursue my dreams and overcome every challenge along the way.

To my advisors, whose mentorship has been invaluable. Your knowledge, dedication, and guidance have shaped not only this research but also my development as a researcher. I am deeply grateful for your insights, constructive feedback, and the trust you placed in my work.

To my incredible work teams, Captação and Shosp, thank you for being a great source of encouragement during the working hours. Your collaboration, shared knowledge, and commitment to excellence have made this experience richer and more meaningful. The synergy and teamwork we have built together have been a source of learning and inspiration.

Lastly, my sincere appreciation to all the individuals who contributed to filling LiMatch. Your participation played a crucial role in the success of this project, and I am truly grateful for your time and effort.

The present work was developed with the support of CAPES - Finance Code 001.

This work is a reflection of the support, encouragement, and dedication of each one of you. Thank you for being part of this journey.

RESUMO

Classificação de Limas Tahiti utilizando Visão Computacional

O limão do Tahiti responde por grande parte das exportações de frutas do Brasil. A classificação da qualidade das frutas é uma questão crítica para o agronegócio, uma vez que o preço da fruta é determinado por sua qualidade. A classificação autônoma de frutas tornou-se um processo essencial, pois permite a seleção rápida e eficiente de frutas para exportação. Essa capacidade reduz custos e aumenta a competitividade no mercado do agronegócio. Neste contexto, foi proposta a criação de um sistema classificador automático de imagens de limas Tahiti que utiliza visão computacional e é capaz de classificar limas utilizando suas características básicas, como cor, e tamanho e possíveis malformações. Para treinar o sistema proposto e avaliar os resultados das técnicas de *machine learning* aplicadas, foi criado um banco de dados contendo centenas de imagens de limas Tahiti dispostas de diferentes maneiras sob diversas condições de iluminação. Para expandir o banco de dados criado, foi aplicado data aumengtation, onde a quantidade de amostras iniciais foi ampliada através da geração de novas amostras através de *Synthetic Minority Oversampling Technique (SMOTE)*. Os resultados obtidos com o modelo proposto atendem aos padrões de qualidade estabelecidos pela *United Nations Economic Commission for Europe (UNECE)* em termos de classificação de maturidade e de avaliação de presença de defeitos. O melhor classificador obteve resultados satisfatórios em termos de maturidade e presença de defeitos, atingindo mais de 90% de precisão usando *Extreme Gradient Boosting (XGB)*.

Palavras-chave: visão computacional,XGB, SMOTE

ABSTRACT

Towards an automatic classification of Tahiti lime

The Tahiti lime accounts for a significant portion of Brazil's fruit exports. The quality classification of fruits is a critical issue for agribusiness, as the price of fruit is determined by its quality. Autonomous fruit classification has become an essential process, as it allows for the rapid and efficient selection of fruits for export. This capability reduces costs and enhances competitiveness in the agribusiness market. In this context, the creation of an automatic classifier system for Tahiti lime images using computer vision has been proposed. This system is capable of classifying limes based on their basic characteristics, such as color, size, and possible deformations. To train the proposed system and evaluate the results of the applied machine learning techniques, a database containing hundreds of Tahiti lime images arranged in different ways under various lighting conditions was created. To expand the created database, data augmentation will be applied, where the initial sample quantity will be increased by using the original image and altering it to obtain another image, or by generating new samples through SMOTE. The results obtained with the proposed model meet the quality standards established by UNECE in terms of file size. The best classifier achieved satisfactory results in terms of maturity and presence of defects, achieving over 85% accuracy using XGB.

Keywords: computer vision, XGB, SMOTE

LIST OF FIGURES

Figure 1 – Proposed lime classification system. 16

Figure 2 – Representation of RGB and HSV spaces. 17

Figure 3 – Dataset image acquisition scheme. 25

Figure 4 – Dataset samples. First row, from left to right: immature and mature limes without defects; second row: immature and mature limes with defects on their skin. 26

Figure 5 – Above: A sample from the dataset. Below: Its HSV color model components (respectively). 28

Figure 6 – The procedure used to generate the lime masks. From left to right, it consists of a blur, thresholding, closing, and AND operation between color image and binary mask. 29

Figure 7 – A masked lime sample with its contour in blue and a red circle circumscribing the contour. The circle's radius represents the lime size. 29

Figure 8 – Screen with register and login fields. 40

Figure 9 – Screen with different limes classes and their definitions. 41

Figure 10 – Screen with different limes classes and their definitions. 41

Figure 11 – Loss and Accuracy, respectively, along the epochs training and validation process for Experiment 5 related with maturity classification. 54

Figure 12 – Loss and Accuracy , respectively, along the epochs training and validation process for Experiment 6 related with the presence of defects classification. 55

Figure 13 – On the left, a representation of a dataset with two classes. On the right, a representation of interpolation method performed by SMOTE technique. 59
Figure 14 – On the left, a representation of a dataset with two classes. On the right, a representation of the instances segmented as Danger, where the SMOTE is applied, becoming a Borderline Smote. 60
Figure 15 – On the left, a representation of a dataset with two classes. On the right, a representation of the instances of the minority class with orange color and the low density regions demarcated within the grey dashed circle, the samples inside of it where used to generate red dots instances. 61
Figure 16 – Organization of the experiments along this work. 68

LIST OF TABLES

Table 1 – Train and Test dataset amounts per classes. 26

Table 2 – Bayesian search performed in the parameter space of the training set using ANN on the Original label. The last two columns are the best parameters for each experiment with respective annotation. 33

Table 3 – Bayesian search performed in the parameter space of the training set using RF on the Original label. The last two columns are the best parameters for each experiment with respective annotation. 35

Table 4 – Bayesian search performed in the parameter space of the training set using XGB on the Original label. The last two columns are the best parameters for each experiment with respective annotation. 36

Table 5 – Performance of classifiers using K-fold with original label. 37

Table 6 – Experiment 1: Mature $\times$ immature limes classification with original label. 37

Table 7 – Experiment 2: With $\times$ without defects limes classification with original label. 38

Table 8 – Number of samples per class in LiMatch's new annotation. 44

Table 9 – Bayesian search performed in the parameter space of the training set using ANN on the Original and LiMatch Label. The last four columns are the best parameters for each experiment, with respective label. 45

Table 10 – Bayesian search performed in the parameter space of the training set using RF on the Original and LiMatch Label. The last four columns are the best parameters for each experiment with respective label. 45

Table 11 – Bayesian search performed in the parameter space of the training set using XGB on the Original and LiMatch Label. The last four columns are the best parameters for each experiment with respective label. 46
Table 12 – Performance of classifiers using K-fold with Original label and LiMatch label. 47
Table 13 – Experiment 3: Mature × immature limes classification with LiMatch label and comparison with Original label. 48
Table 14 – Experiment 4: With × without defects limes classification with LiMatch label and comparison with Original label. 48
Table 15 – Best parameters of the training set using ResNet-152 + ANN with LiMatch label detailed for each experiment. 51
Table 16 – Best parameters of the training set using ResNet-152 + RF with LiMatch label detailed for each experiment. 51
Table 17 – Best parameters of the training set using ResNet-152 + XGB with LiMatch label detailed for each experiment. 52
Table 18 – Performance of classifiers combined with ResNet-152 using K-fold. 54
Table 19 – Experiment 5: Mature × immature limes classification with LiMatch label using ResNet-152. 56
Table 20 – Experiment 6: With × without defects limes classification with LiMatch label using ResNet-152. 57
Table 21 – Best parameters of XGB the training set using SMOTE with LiMatch label for oversampled class. 63
Table 22 – Best parameters of XGB the training set using Borderline SMOTE with LiMatch label for oversampled classes. 63
Table 23 – Best parameters of XGB the training set using ADASYN with LiMatch label for oversampled classes. 64
Table 24 – Number of train samples per class with Data Augmentation methods. 64
Table 25 – Performance of XGB using K-Fold with Limatch Label with oversampled class. 65
Table 26 – Experiment 7: Mature × immature limes classification with LiMatch Label and comparison between data augmentation methods. 66

Table 27 – Experiment 8: With × without defects limes classification with LiMatch Label and comparison between data augmentation methods. 66

Table 28 – Mature × immature limes classification best results. 68

Table 29 – With × without defects limes classification best results. 69

LIST OF ABBREVIATIONS

ADASYN	Adaptive Synthetic
AIM	Angular Invariant Maximal
ANN	Artificial Neural Network
ASM	Angular Second Moment
GLCM	Gray-Level Co-occurrence Matrix
HOG	Histogram Of Oriented Gradients
KNN	K-Nearest Neighbours
PAAS	Platform As A Service
PIC	Pairwise Intensity Comparisons
RESNET	Residual Network
RF	Random Forest
SBRT	Simpósio Brasileiro De Telecomnunicações E Processamento De Sinais
SIFT	Scale-Invariant Feature Transform
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
UNECE	United Nations Economic Commission For Europe
XGB	Extreme Gradient Boosting

CONTENTS

1	**Introduction**	**13**
1.1	Related Works	14
1.2	Contributions	15
2	**Theoretical Foundations**	**16**
2.1	Computational Vision and Image Formation	16
2.2	Segmentation and Processing of an Image	17
2.2.1	Gaussian Filter	18
2.2.2	Binarization	19
2.2.3	Closing Morphology	20
2.2.4	Kurtosis	21
2.2.5	Skewness	22
2.2.6	GLCM Matrix	23
2.3	Metrics	24
2.4	Final Remarks	24
3	**Designed Dataset**	**25**
4	**A fruit classification system**	**27**
4.1	Preprocessing and Feature space mapping with handcrafted features	27
4.2	Classification and Experimental procedures	31
4.2.1	Artificial Neural Network	32
4.2.2	Random Forest	33
4.2.3	Extreme Gradient Boosting	35
4.3	Results and discussion using Original label	36
4.3.1	Results on training and validation sets	36
4.3.2	Results on test dataset using original label	37

5 LiMatch 39
5.1 Experimental procedure applied on LiMatch Label 44
5.1.1 Parameter optimization using LiMatch Label 45
5.2 Results and discussion using LiMatch label 46
5.2.1 Results on training and validation sets 46
5.2.2 Results on test dataset 47
5.3 Preprocessing and Feature space mapping using a convolutional neural network 49
5.3.1 Deep convolutional neural network: ResNet-152 49
5.3.2 Parameter optimization for classifiers combined with ResNet-152 50
5.3.3 Results on training and validation sets using ResNet-152 51
5.3.4 Results on test dataset using ResNet-152 55

6 Data augmentation 58
6.1 Data Augmentation 58
6.1.1 SMOTE 58
6.1.2 Borderline SMOTE 60
6.1.3 ADASYN 61
6.2 Results and discussions using Data Augmentation with LiMatch label 62
6.2.1 Parameter optimization sets for Data Augmentation methos of oversampling classes 63
6.2.2 Results on training and validation sets with oversampled classes 64
6.2.3 Results on test set with Data Augmentation oversampled classes 65

7 Concluding remarks and future work 67
7.1 Future work 70

References 70

1- Introduction

The Persian lime (*Citrus x latifolia*), also known as seedless lime or Tahiti lime, is common in many countries. Citrus production is one of the most significant sectors in Brazilian agribusiness, playing a strategic role in both domestic and international markets. Among the cultivated varieties, the Tahiti lime stands out due to its high commercial acceptance, high productivity, and adaptability to tropical climates (SILVA; ALMEIDA, 2020; OLIVEIRA; SOUZA, 2019). However, the intrinsic variability of these fruits in terms of size, coloration, and ripeness poses a challenge to standardization in production and commercialization. Therefore, precise categorization of these fruits is essential to add value to the final product and ensure competitiveness in the global market (FERREIRA; COSTA, 2021). Efficient and fast fruit classification can improve competitiveness in this sector.

Automated classification of Tahiti limes through computer vision and artificial intelligence emerges as a promising alternative for optimizing the production chain. Traditional methods based on human visual inspection are subjective and prone to inconsistencies, while machine learning-based approaches enable more objective, rapid, and scalable categorization (SANTOS; LIMA, 2022). Moreover, implementing intelligent systems in fruit sorting and selection can reduce post-harvest losses, ensure homogeneous quality standards, and improve traceability—critical factors for meeting international market demands (COSTA; MENDES, 2023).

This work describes a method for autonomous Persian lime classification based on computer vision which was designed to be used under controlled conditions in industrial applications.The purpose of this study is to improve a previous methodology done at (GIACOMETTI et al., 2022) and comparing it with previous results obtained by using Artificial Neural Network (ANN), and Random Forest (RF). In order to achieve this goal a SMOTE data augmentation technique should be used to increase the number of samples. Additionally, the XGB method is implemented to explore comparative model performance. To enhance the performance of the models developed in this work, a tool to improve the label quality of samples was created and defined after Limatch. The main purpose of this tool is to collect as many as possible classifications from several people for each lime

image.

The proposed system in this paper can classify limes according to maturity (related to roughness on the lime skin), presence/absence of defects, and size. We employed three types of classifiers, ANN, XGB and RF. The classification occurs in a feature space that combines size, shape, color, and texture measures. The dataset is composed of images from hundreds of limes acquired in a controlled environment with homogeneous background, constant illumination. We employed data augmentation based on random rotation to increase the data variability and deal with class unbalance issues. The quantitative evaluation of the proposed approach was in terms of accuracy, and the results were promising. However, there is no quantitative comparison with previously published methods because there is no available dataset for Persian lime classification.

1.1- Related Works

The usage of computer vision methods for fruit detection or classification is not a novelty and finds application for different purposes. For example, there are several methods for fruit detection on trees. A method in (LU et al., 2012) that uses a fusion of images segmented based on an adaptive threshold in RGB and HSV color spaces. The method in (POTHEN; NUSKE, 2016) uses Angular Invariant Maximal (AIM) to detect key points, and then Pairwise Intensity Comparisons (PIC) and gradients (based on Histogram of Oriented Gradients (HOG)) are used to extract a feature map. The RF technique performs the classification step.

There are many fruit classification methods as well. The work in (ZAWBAA et al., 2014) compared the color and shape with Scale-Invariant Feature Transform (SIFT) as features. They compared three methods in the classification step: RF, k-Nearest Neighbours (KNN), and Support Vector Machine (SVM).

A method for tomato detection of defects was employed by (DA COSTA; FIGUEROA; FRACAROLLI, 2020) using a commercial structure for sorting with regular camera chamber and fluorescent illumination to acquire more than 40,000 images.

They employed several techniques for classification: SVM, k-NN, Classification and Regression Tree (CART), Linear Discriminant Analysis (LDA), and Fisher Discriminant

Analysis (FDA) are the most commonly applied algorithms. An approach to localize Emperor citrus using RGB-D (RGB and depth) information with SVM is presented in (LIN et al., 2019). There are some methods for detecting defects in citrus fruit in the literature. Although the work in (BLASCO et al., 2009) also deals with defects, it aims to classify among 11 different types of defects, differently from others whose object is to discriminate between healthy and defected fruits.

1.2- Contributions

The work has some main contributions:

1. A lime classification system;
2. A Tahiti lime dataset, exclusively designed for this task, available at a private GitHub, providing a dataset for numerous other purposes and filling a gap, as there are no publicly available image datasets for Tahiti limes up to the time this text is being written;
3. A tool to improve datasets' labeling process;
4. A straight forward comparison between a deep convolutional neural network and others classic algorithms;
5. An analysis of how different methods of SMOTE impacts accuracy.

The study conducted around the topic resulted in the publication of a main scientific research product called Towards an Automatic Classification of Persian Lime (GIACOMETTI et al., 2022) which was published at the *Simpósio Brasileiro de Telecomnunicações e Processamento de sinais (SBrT)*.

2- Theoretical Foundations

The purpose of this section is to clarify the main concepts that will be applied in the process to elaborate the methodology proposed in Figure 1. The concepts addressed in Section 2.1 refer to how an image is formed, Section 2.2 is about the processes applied to the image to enhance the main characteristics of lime that will be treated as features and will be processed, and then evaluated using the metrics explained on Section 2.3.

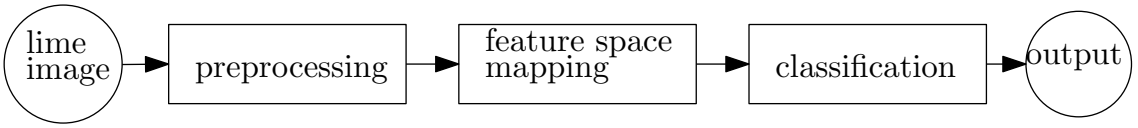


Figure 1 – Proposed lime classification system.

In case you are familiar with the fundamental concepts of computational vision and machine learning, you should consider moving on to Chapter 4.

2.1- Computational Vision and Image Formation

Computer vision is a complex field with widespread applications in our society. It has been used for many years in various fields such as industrial inspection and mobile robot navigation (FORSYTH; PONCE, 2003). Conceptually, computer vision can be defined as the process of interpreting explicit and meaningful attributes of physical objects from images (BALLARD; BROWN, 1982). In a broader context, computer vision is an interdisciplinary field based on the fields of image processing and machine learning. It covers a wide range of topics, from image acquisition, processing, and segmentation to classifying objects in images (FORSYTH; PONCE, 2003).

A strong argument in favor of initiating the integration of computer vision into a project lies in the initial cost efficiency associated with acquiring images. When compared against the substantial benefits conferred by the application of this methodology, the relatively low cost of image acquisition becomes an attractive proposition (FORSYTH;

PONCE, 2003).

2.2- Segmentation and Processing of an Image

Color images are frequently represented as combinations of channels, especially when they are produced by digital systems. These channels are roughly divided into the Red (R), Green (G), and Blue (B) channels in the RGB color space based on the wavelength ranges to which sensors are sensitive (ACHARYA; RAY, 2005). A cube is frequently used as a geometrical representation of the RGB color space, with its axes designating the extreme ends of the red, green, and blue color components. The HSV color space, on the other hand, is represented as a cone, where hue is determined by the angular change along the perpendicular axis that represents the value attribute and saturation varies linearly along the apothem of the hexagon (FORSYTH; PONCE, 2003) as could be seen in Figure 2.

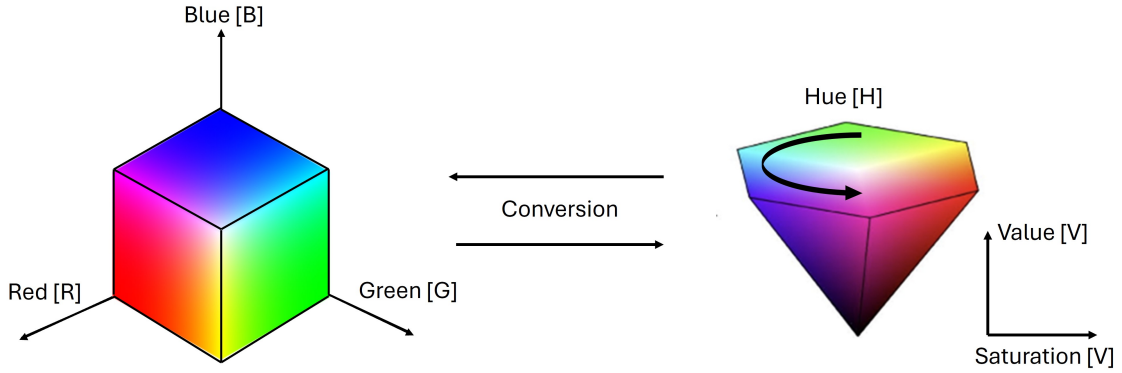


Figure 2 – Representation of RGB and HSV spaces based on (CHERNOV; ALANDER; BOCHKO, 2015) reference.

In this work, we used a conversion between color spaces this transformation is beneficial for segmentation algorithms, as it isolates the color component from the intensity and lighting variations. In the HSV space, for instance, it is easier to segment a specific color, such as green, by adjusting the hue while keeping the saturation and value constant, a process that can be more complex in RGB due to the interdependence of its components.To accomplish this conversion first it is necessary to normalize RGB channels

through the following equations (1):

$$\begin{aligned} R' &= R/255, \\ G' &= G/255, \\ B' &= B/255. \end{aligned} \tag{1}$$

Then, $C_{\max}$ is defined by the highest normalized value and $C_{\min}$ as the lowest one as represented by the equations (2):

$$\begin{aligned} C_{\max} &= \max(R', G', B'), \\ C_{\min} &= \min(R', G', B'), \end{aligned} \tag{2}$$

then it is defined delta as described in equation (3):

$$\Delta = C_{\text{max}} - C_{\text{min}}. \tag{3}$$

The function that describes the conversion between RGB to HSV space is defined as described in equation (4), where $(\mathrm{mod}\ 6)$ is the rest of a division by 6.

$$\begin{aligned} H &= \begin{cases} 60 \cdot \left(\dfrac{G' - B'}{\Delta} \pmod 6 \right), & C_{\text{max}} = R' \\ 60 \cdot \left(\dfrac{B' + R'}{\Delta} + 2 \right), & C_{\text{max}} = G', \\ 60 \cdot \left(\dfrac{R' - G'}{\Delta} + 4 \right), & C_{\text{max}} = B' \end{cases} \\ S &= \begin{cases} 0, & C_{\text{max}} = 0 \\ \dfrac{\Delta}{C_{\text{max}}}, & C_{\text{max}} \neq 0, \end{cases} \\ V &= C_{\text{max}}. \end{aligned} \tag{4}$$

2.2.1 Gaussian Filter

In the context of image processing, filtering involves applying transformations to the image in a manner that renders the discontinuities between region segments more

pronounced or smoothed according to the intended purpose (BALLARD; BROWN, 1982). There are various types of filters, such as Gaussian filter, Morphological filter, Median Filter, etc (KRIZHEVSKY; SUTSKEVER; HINTON, 2017).

The Gaussian Filter is used to enhance, smooth and denoise the image. This filter operates based on the convolution of a kernel described by a Gaussian function with the pixels of the image (GEDRAITE; HADAD, 2011).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \tag{5}$$

Equation (5) could be seen a representation of the Gaussian function in 2D where x is a gray level image and μ is the mean value, and σ is the standard deviation which determines the amount of smoothing (KUMAR; SODHI, 2020).

This operation in the discrete model is given by Equation (6), which describes a convolution where m and n are the dimensions of the image in question:

$$y(m,n) = \sum_{k,l=-\infty}^{\infty} x(k,l) \cdot h(m-k, n-l). \tag{6}$$

The Gaussian distribution provides values that are utilized in the creation of a convolution matrix. This matrix is subsequently employed in the processing of the original image. The computation of each new pixel value involves a weighted average of its neighboring pixels. Consequently, the original pixel, with the highest Gaussian value, is assigned the greatest weight, while the surrounding pixels contribute with progressively lower weights as the distance from the original pixel increases.

2.2.2 Binarization

Binarization is a technique for converting the pixel image into the binary image (INGLE; KAUR, 2017). To implement this technique, it is essential for the image to be in grayscale, where each pixel has the luminosity of that particular area. Following this, a threshold is established. The application of this threshold to the image involves setting pixels with values below the threshold to zero, and those equal to or exceeding the threshold to one. Consequently, this emphasizes specific regions in the image, delineating distinct

segments (BRADSKI, 2000).

2.2.3 Closing Morphology

Closing morphology is a crucial operation in the field of image processing. This technique is essential for noise reduction, shape refinement, and image segmentation. With it is possible to fill small holes within foreground objects in binary images, not only that but also helps eliminate small dark spots (often referred to as ”pepper” noise) from images (SERRA, 1982).

There are several techniques of morphology, and the order of the processes applied may affect the final result. The dilation morphology, followed by the erosion morphology results on closing morphology applied in this work.

The dilation operation involves a union of all sets obtained from translations of the image pixels by the structuring element. This process allows for the expansion of the white region, effectively eliminating noise such as dark spots within the lime contour caused by insufficient light incidence on the peel. Otherwise, the morphological erosion operation can be understood as the intersection of all sets obtained from translations of image pixels through the reflection of the structuring element. In this study, erosion was employed to vanish the edges that delineate the background and the object. In this operation, a disk-shaped structuring element is created, sliding through the image and applying the convolution operation. Thus, a pixel in the original image is considered 1 only if the resulting pixel from the kernel is 1; otherwise, it undergoes erosion, assuming a null value. The erosion effect results in the loss of most pixels near the edge, and the quantity of these eroded pixels depends on the size and shape of the structuring element.

These morphological operations play a crucial role in image processing, providing mechanisms to enhance or diminish specific features within an image (SZELISKI, 2010).

2.2.4 Kurtosis

Kurtosis is a statistical measure that helps us understand the shape of a probability distribution and also provide insights into the behavior of distributions, particularly regarding extreme values. Therefore, datasets characterized by high kurtosis tend to exhibit outliers. These outliers represent extreme values that are significantly different from the majority of the data points. In contrast, a dataset with low kurtosis tends to have a lower number of outliers, indicating a distribution with less variability and fewer extreme values (JOANES; GILL, 1998). Additionally, by definition, kurtosis represents the fourth central moment divided by the square of the variance. The Equation (7) defines the kurtosis of a dataset as:

$$a_4 = \frac{m_4}{m_2^2}, \tag{7}$$

where:

$$m_4 = \frac{\sum(x-\bar{x})^4}{z}, \text{ and}$$
$$m_2 = \frac{\sum(x-\bar{x})^2}{z},$$

being $\bar{x}$ the average and z the sample size, m_4 is known as the fourth central moment of a dataset, m_2 is the sample variance (NAVIDI, 2010). Because of the sample size z being on the denominator of this equation , its possible to affirm that both m_2 and m_4 are biased.

Kurtosis can be applied to the pixel intensity values within local regions or neighborhoods of an image to characterize the texture properties. The presence of texture changes or features is suggested by the high kurtosis values in certain places, which show that the pixel intensity distribution has heavier tails. These regions are likely to contain textured areas with abrupt changes in pixel intensity, such as edges or patterns. On the other hand, low kurtosis values suggest homogeneous regions with less texture variation by indicating a smoother distribution of pixel intensities. These areas could be the image's smooth surfaces or areas with consistent color (HARALICK; SHANMUGAM; DINSTEIN, 1973). For the actual issue these feature are used to treat the lime image so the only thing that would be processed is really the reagion which represents the lime.

2.2.5 Skewness

Skewness quantifies how much a set of samples deviates from a perfectly symmetrical distribution. It is a measure of symmetry, or more accurately, of assymmetry. If the density samples graph looks the same when viewed from both the left and the right up to the central point, it is said to be symmetric.

The third standardized moment of the distribution is used to calculate skewness. Skewness is zero in a distribution that is entirely symmetrical. When data is skewed, it can be either positive or negative. Positive skewness indicates a tendency to the right. Furthermore, negative skewness indicates a tendency to the left (TABACHNICK; FIDELL, 2021).

When it comes to the pixel domain, positive skewness indicates that there may be a concentration of lower intensity values along with a small number of higher intensity values in areas where the pixel intensity distribution is positively skewed. Positive skewness, for example, may be seen in areas with noticeable darker areas and rough textures. On the other hand, negative skewness indicates that there is a concentration of higher intensity values with a few lower intensity values in areas where the pixel intensity distribution is negatively skewed. This may suggest the existence of textures where the pattern's general color is dominated by lighter or brighter areas. For example, areas with smooth textures and noticeable lighter spots could show negative skewness (**haralick1973textural**).

The Equation (8) defines the skewness of a dataset as:

$$g_1 = \frac{m_3}{m_2^{\frac{3}{2}}}, \tag{8}$$

where:

$$m_3 = \frac{\sum(x-\bar{x})^3}{z}, \text{ and} \tag{9}$$

$$m_2 = \frac{\sum(x-\bar{x})^2}{z}, \tag{10}$$

being $\bar{x}$ the average and z the sample size, m_3 is known as the third central moment, m_2 is the sample variance (NAVIDI, 2010).

2.2.6 GLCM Matrix

Haralick proposed the Gray-Level Co-occurrence Matrix (GLCM), which is today a popular technique for extracting texture information from photos (**haralick1973textural**). Its most frequent description is that it is a statistical method used to study the spatial correlations between pixel values in an image, which could be translated into counting the frequency of different grayscale levels that pixels display in an image.

In this work, the following texture properties were extracted from a GLCM using the Scikit-Learn library (PEDREGOSA et al., 2011).

- Contrast: measures the local changes in pixel intensity. High contrast ratings indicate sharp differences in pixel values in the image.
- Homogeneity: is a metric that measures how close the distribution of elements in the GLCM is to its diagonal. A higher homogeneity score indicates that the visual texture is more uniform.
- Energy: also known as Angular Second Moment (ASM), is a measure of the local homogeneity of visual texture. Higher energy values correspond to a more uniform texture.
- Correlation: determines the linear relationship between pixel pairings in the GLCM. It reflects how closely connected the pixel values in the image are.
- Entropy: is a measure of the unpredictability or uncertainty of pixel pairs in the GLCM. Higher entropy levels suggest a more complex texture.
- Dissimilarity: measures the average difference between pixel intensities in the image. Higher dissimilarity values indicate greater variations in pixel intensities.

The equations and applications of these features will be discussed on Section 4.1

2.3- Metrics

In this section, it introduces the metrics used in this work. It is usual to evaluate classification problems by metrics derived from the confusion matrix. This work used *Scikit-Learn* (PEDREGOSA et al., 2011) to generate them for each experiment. Another evaluate measure used is balanced accuracy displayed on Equation (11), applied because classes are not balanced regarding the number of samples for each class.

$$\rho = \frac{1}{2}\left(\frac{T_P}{T_P + F_N} + \frac{T_N}{T_N + F_P}\right). \tag{11}$$

At the balanced accuracy (ρ) formula T_P means true positive, T_N means true negative, F_P denotes false positive and F_N computer the number of false negative, and it was applied for the same reason stated before.

2.4- Final Remarks

All the concepts discussed in this chapter will be practically applied in the following sections, where the image processing steps performed on lime images will be explained and illustrated with examples. These practical applications are further detailed in Chapter 3.

3- Designed Dataset

This chapter is dedicated to elucidating the procedural aspects of image base creation and the subsequent application thereof. Within the classifier development process lies a crucial stage known as the training phase. During this phase, each classifier is furnished with attributes derived from a substantial pool of samples. These samples are systematically labeled in accordance with their respective classifications. Subsequently, employing statistical methodologies, the classifier correlates these labels with the corresponding classifications.

The designed dataset is composed of images from 1083 limes in several sizes and conditions. Each image has 793×793 pixels dimension. The acquisition occurred in a controlled environment with homogeneous background, constant illumination, and an image plane parallel to the background at a fixed distance, according to Figure 3.

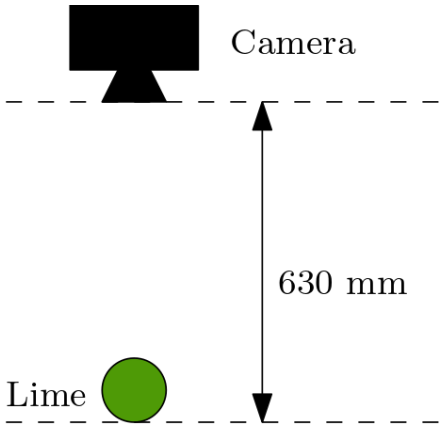


Figure 3 – Dataset image acquisition scheme.

Each image from the dataset was labeled as immature or mature and as with or without defects. This annotation process was performed initially by only one person, who was taught to avail limes according to UNECE standards and them revised by another person. As can be seen in Figure 4, immature Tahiti limes have a rough dark green aspect. Mature limes are smooth, green, and can be yellowish in some parts. A lime with defects has brown features in its skin and can be mature or immature.

The dataset has two parts, the training, and test subsets. The training is composed of 959 and the test of 124 images. The amount of images, defined for each group test and



Figure 4 – Dataset samples. First row, from left to right: immature and mature limes without defects; second row: immature and mature limes with defects on their skin.

training, was designed to set the training group as huge as possible without negatively impact the number of images atrributed to test subset. The training set contains 287 immature without defects, 72 immature with defects, 300 mature without defects, and 300 mature lime images with defects. The test set has 20 immature without defects, 24 immature with defects, 42 mature without defects, and 38 mature limes with defects. As one can note, the classes are highly unbalanced. These organization could be checked at Table 1.

Table 1 – Train and Test dataset amounts per classes.

Classes	Amount Samples	
	Train	Test
Mature (L)	600	80
Not Mature (R)	359	44
Defective (C)	372	62
Not Defective (S)	587	62
Total	1918	248

4- A fruit classification system

A block diagram of the proposed system was in Figure 1. It is a straightforward three-step system composed of a pipeline of preprocessing, feature space mapping, and classification. A more detailed description of each step follows in the next sections.

4.1- Preprocessing and Feature space mapping with handcrafted features

This section describes the preprocessing and feature scheme used in the proposed system. The ultimate goal of the image processing pipeline is to completely isolate the lime from any other elements present in the image, such as shadows or the background. This ensures that the extracted features are focused entirely on the lime itself, improving the accuracy and reliability of later classification steps. To achieve this, multiple empirical tests were conducted to optimize the differentiation between lime and other components. Key parameters, including the degree of blur, the binarization threshold, and the shape and size of the structuring element used in the morphological closing operation, were defined through manual adjustments and subjective evaluation of the results. The first step is the color conversion from the original RGB color space to HSV model. By doing this it is possible to obtain the color range from channel of saturation, which probably would prove to be an important feature for the model. All images in this work are floating-point variables, so each pixel in each channel is in the range $[0, 1]$. This procedure is in Figure 5 which shows the original RGB image $\mathbf{I}$ of a lemon image and its respective $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{V}$ channels.

Figure 6 illustrates the masking operation. One can note that the $\mathbf{S}$ channel emphasizes the lime area from the background. The feature mapping occurs in this area, so we perform masking as follows. A Gaussian filter was applied to smooth any high frequency noise that may be presented in the image. Let $\mathbf{G}$ be a Gaussian filter with



Figure 5 – Above: A sample from the dataset. Below: Its HSV color model components (respectively).

standard deviation σ from Equation (12),

$$\sigma = 0.3(0.5(K-1)-1)+0.8, \tag{12}$$

with kernel size $K \times K$ pixels ($K = 5$ in this work). Equation (12) is described in the manual of *getGaussianKernel* OpenCV function and is used to set the standard deviation applied to the Gaussian filter. We can obtain the blurred channel $\mathbf{S_b}$ from a simple convolution.

Then we binarize the blurred S channel, $\mathbf{S}_b$ by employing a threshold of 50% of the range. To fill the holes, gaps and remove saliences we use a morphological closing operation in the binarized image $\mathbf{S}_t$ with a disk-shaped structuring element $\boldsymbol{b}$ in a $B \times B$ pixels kernel ($B = 30$ in this work), resulting in the desired mask $\mathbf{\Delta}$. The closing operation can be obtained by employing a dilation followed by an erosion operation.

We obtain the region of interest RoI through AND logical operation, between the mask and the original image. The full process is shown in Figure 6.

We perform the feature space mapping within the Region of Interest (RoI) of each sample, as shown in Figure 7. The *findContours* OpenCV function compute the blue contour. Another OpenCV function employed was *minEnclosingCircle*. It was responsible for generating the red circle that circumscribes the contour. The first feature is the diameter



Figure 6 – The procedure used to generate the lime masks. From left to right, it consists of a blur, thresholding, closing, and AND operation between color image and binary mask.

D, which we use as the lime size measurement. We estimate the diameter through Equation (13),

$$D = 2(0.21)R, \tag{13}$$

where R is the radius of the circle given in [pixels] obtained by minEnclosingCircle, 0.21 is an experimentally obtained conversion constant given in $\left[\frac{\text{cm}}{\text{pixels}}\right]$ and D is the diameter in [cm]. To determine the conversion factor described here as 0.21 , an image was captured

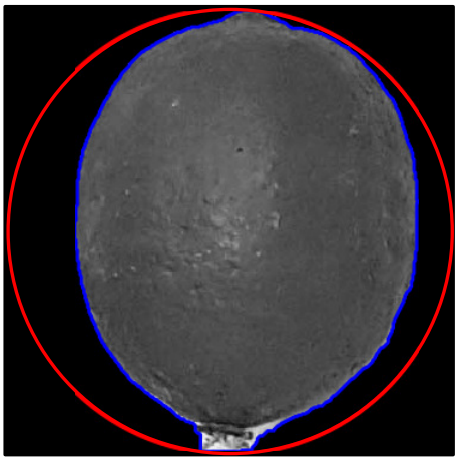


Figure 7 – A masked lime sample with its contour in blue and a red circle circumscribing the contour. The circle's radius represents the lime size.

under the same controlled conditions described earlier for dataset acquisition. The image contained a lime alongside a metric ruler. By analyzing the image, the relationship between the number of pixels and one centimeter was calculated, resulting in a conversion ratio of 0.21 pixels per centimeter.

We obtain the remaining features through the $\mathbf{H}$ and $\mathbf{V}$ channels of the RoI. The second feature is the histogram $\mathbf{h_H}$ of the $\mathbf{H}$ channel.

The texture features are measurements from the $\mathbf{V}$ channel. They are Skewness s_k, Kurtosis k_u and the following GLCM $\mathbf{P}$ features: contrast c_n, dissimilarity d_i, homo-

geneity h_o, Angular Second Moment (ASM) a, energy e_n and correlation c_r. The texture features are defined in Equations (14) to (21) and the feature vector $\mathbf{x}$ is composed by the concatenation of the described features as in Equation (22). All the features were obtained through *scikit* library (PEDREGOSA et al., 2011), which are based on (HARALICK; SHANMUGAM; DINSTEIN, 1973).

The detailed equations are:

$$\nu_r = \frac{1}{MN} \sum_{\forall(m,n)} (\mathbf{V}(m,n) - \mu_{\mathbf{V}})^r, \text{ for } r = 2, 3, \text{and } 4, \tag{14}$$

$$s_k = \frac{\nu_3}{\nu_2^{3/2}}, \tag{15}$$

$$k_u = \frac{\nu_4}{\nu_2^2}, \tag{16}$$

$$c_n = \sum_{\forall m,n} (m-n)^2 \mathbf{P}(m,n), \tag{17}$$

$$h_o = \sum_{\forall m,n} \frac{\mathbf{P}(m,n)}{1+(m-n)^2}, \tag{18}$$

$$a = \sum_{\forall m,n} \mathbf{P}^2(m,n), \tag{19}$$

$$e_n = \sqrt{a}, \tag{20}$$

$$c_r = \sum_{\forall m,n} \mathbf{P}(m,n) \left[\frac{(m-\mu_m)(n-\mu_n)}{\sigma_m \sigma_n} \right], \text{ and} \tag{21}$$

$$\mathbf{x} = [\mathbf{h}_H, s_k, k_u, c_n, h_o, a, e_n, c_r], \tag{22}$$

where $\mu_{\mathbf{V}} = \frac{1}{MN} \sum_{\forall(m,n)} (\mathbf{V}(m,n))$, M is the horizontal number of pixels of the image, N is the vertical number of pixels of the image, $\mu_m = \frac{1}{N} \sum_{\forall n} (\mathbf{P}(m,n))$ for $m = 1, \ldots, M$, and $\mu_n = \frac{1}{M} \sum_{\forall m} (\mathbf{P}(m,n))$ for $n = 1, \ldots, N$.

The next step of the system is the classification, which is performed in a feature space that combines color and texture measurements.

4.2- Classification and Experimental procedures

This work addresses two different types of classification to evaluate the capability of different classifiers to categorize limes according to their maturity and their quality related to presence or absence of defects. The classifiers employed were ANN, RF and XGB. These classifiers were selected for the actual relevance found in he recent academic papers (CHEN, K., 2023). The time performance related to the time spent to train the models was considered as a key point. There are two scenarios for each type of classifier, one to distinguish between a mature and an immature lime (experiment 1), and the other to classify a lime from others without defects (experiment 2). All algorithms and parameters applied for each classifier used are available on Scikit-learn Python library (PEDREGOSA et al., 2011).

Datasets have variability, which means they can vary in terms of the distribution of data points, which can affect the performance of machine learning models, especially if the classes are unbalanced, which is the case of the dataset that was developed for this study and was detailed in Chapter 3.

Cross-validation is a machine learning technique for evaluating the performance of a prediction model. It is especially effective when the dataset is restricted since it allows for the most efficient use of available data for training and testing.

In this work, a stratified K-Fold cross-validation from Scikit-learn library (PEDREGOSA et al., 2011) was applied using 5 folds. The performance results of this application itself are detailed in Section 4.3.1. The application of a stratified K- fold cross validation and a Bayesian search lead to the best hyperparameters picked among the sample space defined. The hyperparameters used for each classified are described in the following sections and the best results selected for this work are described at Table 2, Table 3 and Table 4.

4.2.1 Artificial Neural Network

An Artificial Neural Network (ANN) is an information processing system inspired by the structure and function of biological neural networks. Despite drawing conceptual parallels to brain neural networks, ANNs differ significantly in their operational processes and implementations (SCHMIDHUBER, 2015).

Three primary types of ANN architectures are widely discussed in the literature: single-layer feedforward networks, multi-layer feedforward networks, and recurrent networks (GOODFELLOW; BENGIO; COURVILLE, 2016). This work employs a multi-layer feedforward network, commonly referred to as a Multi-Layer Perceptron (MLP), combined with the backpropagation learning algorithm. The MLP is particularly effective in recognizing and classifying linearly non-separable patterns. Backpropagation, initially introduced by Werbos (WERBOS, 1975) and popularized in subsequent research, is a supervised learning technique that iteratively minimizes the network error by adjusting weights, making it foundational for modern ANN training.

Bayesian Search Parameters for ANN

The parameters selected for use in the Bayesian Search of the ANN classifiers are as follows:

- ***Hidden Layer Sizes***: This represents the number of neurons in each hidden layer (PEDREGOSA et al., 2011).
- ***Activation***: It denotes the activation function for the hidden layers. In this study, it can be either a hyperbolic tangent function (*tanh*) or a rectified linear unit function (*relu*) (PEDREGOSA et al., 2011).
- ***Alpha***: This is the R2 regularization term (PEDREGOSA et al., 2011).
- ***Solver***: It refers to the optimizer for weight optimization, where *lbfgs* is a Quasi-Newton optimizer, *sgd* denotes stochastic gradient descent, and *adam* refers to a gradient-based optimizer (PEDREGOSA et al., 2011).

- ***Learning Rate***: This is the learning rate used to update the weight values. It can be set as a constant at a value of 0.001 (*constant*) or be adaptive, maintaining the rate at 0.001 as long as the model continues to converge (PEDREGOSA et al., 2011).
- ***Random State***: This is the value passed to the pseudo-random number generator (PEDREGOSA et al., 2011).

This information provides a comprehensive overview of the parameters considered for tuning the artificial neural network (ANN) classifiers, results using this technique can be seen in Table 2.

Table 2 – Bayesian search performed in the parameter space of the training set using ANN on the Original label. The last two columns are the best parameters for each experiment with respective annotation.

Parameter	**Bayesian Search**	**Original Label**	
		Exp 1	**Exp 2**
Neurons in hidden layer	2,3, $\cdots$, 500	500	500
Activation function	('tanh', 'relu')	relu	relu
Optimization algorithm	('sgd', 'adam','lbfgs')	adam	adam
ℓ_2 penalty (regularization term α)	1^{-6} ,1^{-5}, 0.0001, 0.05, 1	1^{-6}	1
Learning Rate	('constant', 'adaptive')	constant	adaptive
Random State	1, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80,90,100	100	1

4.2.2 Random Forest

RF is a decision-tree-based machine learning algorithm used in this case to solve a classification. This method uses bagging for creating multiple models (decision trees) from a single training dataset, resulting in a very powerful classifier developed and described by (BREIMAN, 2001) and (JAMES et al., 2013).

Bayesian Search Parameters for RF

The parameters selected to be utilized in the Bayesian Search of Random Forest classifiers were:

- ***N estimators***: This refers to the number of trees used by the classifier (PEDREGOSA et al., 2011).
- ***Max features***: It denotes the number of attributes to be considered for the best split. It can be automatic, denoted by the option *auto*, or determined by the square root of the number of features, denoted by the option *sqrt (PEDREGOSA et al., 2011)*.
- ***Max depth***: This is the maximum depth reached by each tree (PEDREGOSA et al., 2011).
- ***Min samples split***: It is the minimum number of samples required before an internal node is split (PEDREGOSA et al., 2011).
- ***Min samples leaf***: This is the minimum number of samples required to form a leaf (PEDREGOSA et al., 2011).
- ***Bootstrap***: It defines whether samples will be used during tree construction. If the option *False* is selected, all samples will be used to build each tree (PEDREGOSA et al., 2011).

This explanation elucidates the various parameters considered for tuning the classifier, the Table 3 shows the results of applying Bayesian Search on a RF classifier.

In general, forests with trees maximum depth above 5 can lead to overfitting (a model too specialized in the training set with no generalization capability) (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). However, it did not occur in the results, as can be seen in Section 4.3.

In general, increasing the maximum depth of trees beyond a certain point, such as 5, can increase the risk of overfitting. This is because deeper trees tend to capture more specific patterns from the training data, which could not generalize well to unseen data. However, the extent to which overfitting occurs depends on factors such as the size and diversity of the dataset, as well as the regularization techniques applied (BREIMAN, 2001) (CARUANA et al., 2008).

Table 3 – Bayesian search performed in the parameter space of the training set using RF on the Original label. The last two columns are the best parameters for each experiment with respective annotation.

Parameter	Bayesian Search	Original Label	
		Exp 1	Exp 2
Number of trees	$20, 30, \cdots, 2000$	406	1560
Maximum number of features at every split	auto, sqrt , log	sqrt	auto
Maximum tree depth	$10, 11, \cdots, 30$	18	22
Minimum number of samples to split	$2, 5, 10, 15$	2	2
Minimum number of samples at each node leaf	$1, 2, 4, 6$	2	1
Bootstrap	true, false	false	false

4.2.3 Extreme Gradient Boosting

XGBoost or XGB, short for Extreme Gradient Boosting, is an implementation of the gradient boosting framework, which improves upon weak learners (typically decision trees) by sequentially building new models that correct the errors of previous ones. For binary or multi-class classification tasks, XGBoost optimizes the logistic loss function, and the model's output is the probability of a sample belonging to a particular class. The predicted probability is then thresholded to assign class label (CHEN; GUESTRIN, 2016),(CHEN, T., 2023).

Bayesian Search Parameters for XGB

The parameters selected for use in the Bayesian Search of the XGB classifier are as follows:

- ***N estimators***: This represents the number of trees to be built (PEDREGOSA et al., 2011).
- ***Max depth***: The maximum depth of each decision tree (PEDREGOSA et al., 2011).
- ***Learning Rate***: Controls how much each tree contributes to the final model. A lower learning rate often requires more trees (PEDREGOSA et al., 2011).

- ***Subsample***: The fraction of the training data used to grow each tree (PEDREGOSA et al., 2011).
- ***Colsample by tree***: The fraction of features used to build each tree (PEDREGOSA et al., 2011).

This information provides a comprehensive overview of the parameters considered for tuning the XGB classifiers, results using this technique can be seen in Table 4.

Table 4 – Bayesian search performed in the parameter space of the training set using XGB on the Original label. The last two columns are the best parameters for each experiment with respective annotation.

Parameter	**Bayesian Search**	**Original Label**	
		Exp 1	**Exp 2**
Number of Trees	$10, \cdots, 200$	153	29
Max Depth	$1, \cdots, 50$	15	15
Subsample	$0.5, \cdots, 1$	0.85	0.91
Learning Rate	$0.01, \cdots, 0.3$	0.3	0.23
Colsample by tree	$0.5, \cdots, 1$	0.77	0.82

4.3- Results and discussion using Original label

In this section, we present and discuss some results of applying the models selected in Section 4.2. At the Section 4.3.1 the K-fold accuracy is listed and the results obtained using it on a test dataset, described in Chapter 3, are described at Section 4.3.2.

4.3.1 Results on training and validation sets

As was briefly stated before a stratified K-fold method was applied during Bayesian search to compare the performance of different models and select the best one choosing it by its average accuracy. The average accuracy and their respective standard deviation obtained during the training process for each experiment and classifier were stated in

Table 5.

Table 5 – Performance of classifiers using K-fold with original label.

Classifier	Experiment	Label	Accuracy ρ [%]
Random Forest	Exp 1	Original	83.86 ± 1.99
	Exp 2	Original	82.81 ± 1.87
ANN	Exp 1	Original	68.75 ± 2.97
	Exp 2	Original	72.77 ± 2.39
XGB	Exp 1	Original	83.48 ± 1.82
	Exp 2	Original	82.08 ± 1.61

A good reason to apply a stratified K-fold technique is that it is possible to mitigate over fitting issues by training and testing the model on different subsets of the data. In this work the standard deviation is small which is a very likely sign of good generalization on unseen data for this model.

4.3.2 Results on test dataset using original label

The accuracy of the classification systems concerning the maturity and defects of the limes, here called Experiment 1 and Experiment 2, respectively, are in Tables 6 and 7.

Table 6 – Experiment 1: Mature $\times$ immature limes classification with original label.

Classifier	T_P	F_P	T_N	F_N	ρ [%]
ANN	57	23	21	23	64.52
RF	74	6	15	29	83.06
XGB	69	11	14	30	79.83

Table 6 presents the performance for the maturity classification related to Experiment 1. The classes mature and immature limes present 80 and 44 samples, as was displayed in Table 1. The classifiers ANN and XGB achieved 64.52%, and 79.83% of hit rate on the test set, respectively, while RF performed 83.06% with the same configurations, showing the performance superiority of the RF for the problem under analysis. It is worth noticing that the sensitivity for mature lime class $\left(\frac{T_P}{T_P+F_N}\right)$ was 71.84% using RF, which is an interesting metric that clarifies the ability of the model to determine the limes that are mature among all of them.

Table 7 – Experiment 2: With $\times$ without defects limes classification with original label.

Classifier	T_P	F_P	T_N	F_N	$\rho\,[\%]$
ANN	17	45	4	58	60.48
RF	43	19	7	55	79.03
XGB	38	24	8	54	74.19

Table 7 presents the performance for the defect classification in the limes. Different from the maturity classification, where the classes differ by almost twice the number of images, the defect classification presents 62 images with and 62 images without defects classes, which represent practically the same amount as can be seen in Table 1. The classifiers ANN, and XGB achieved 60.48%, and 74.19% of hit rate on the test set, respectively, while RF reached 79.03% showing a slight improvement in using the RF classifier. It is worth noticing that the highest specificity, $\left(\frac{T_N}{T_N+F_P}\right)$, was 26.92%, also using RF, that means that the classifier still have opportunity to identify better true negatives cases.

Despite the reasonable results achieved using RF classifiers, the classifier might have been biased by the annotation process done. To reduce this probability it was proposed a annotation system detailed in Chapter 5.

5- LiMatch

The process of building a solid annotation requires a human to classify each lime presented in the images taken. To acknowledge this mission a human needs to read about limes and their stages of ripeness; only after that, is possible to deduce correctly a class for each lime in the image. After a few analyses, was clear that the labeling process was subjective and vary depending on who was classifying limes. To minimize the detected human bias, a system called LiMatch was proposed. It was developed with the intention of training people about how to classify limes and then asking them to annotate some limes images.

LiMatch's first version was developed as a standalone application solution. Although there were several people involved with the project and assigned to use LiMatch to learn and classify limes, was clear that this type of system needs to be a web application to reach out to as many people as possible generating an increase in the number of classifications and enlarge the number of people with different skills who analyze the limes.

So Limatch was turned into a web application that uses user input to speed up the image classification process. It is built on the Django framework. Its primary role is to let users categorize photographs into two of five pre-established groups. By utilizing Django's (HOLOVATY; KAPLAN-MOSS, 2009) strong characteristics for scalability and fast development, Limatch offers a smooth and user-friendly interface for users to engage with the classification activities.

Limatch's architecture is based on the Model-View-Controller (MVC) pattern found in Django. Models specify the database schema that stores image data and labels for classification, views manage user requests and generate dynamic webpages, and templates supply CSS and HTML markup for the user interface. This architecture ensures modularity, maintainability, and extensibility, allowing for easy integration of additional features and customization options in the future.

Limatch's categorization process is simple to use: after viewing an image, viewers are asked to choose from a list of pre-established categories that most accurately characterize the image's content. Form submission and validation are made easier by

Django's form handling features, which guarantee that user inputs follow the designated options and constraints. However, Limatch is a useful tool for obtaining labeled data to train machine learning models because of its emphasis on efficiency and simplicity, especially in situations when manual annotation is required or desired.

Three pages were developed to accomplish the system's goal. The first one was developed to get information about who is analyzing limes images as shown in Figure 8.

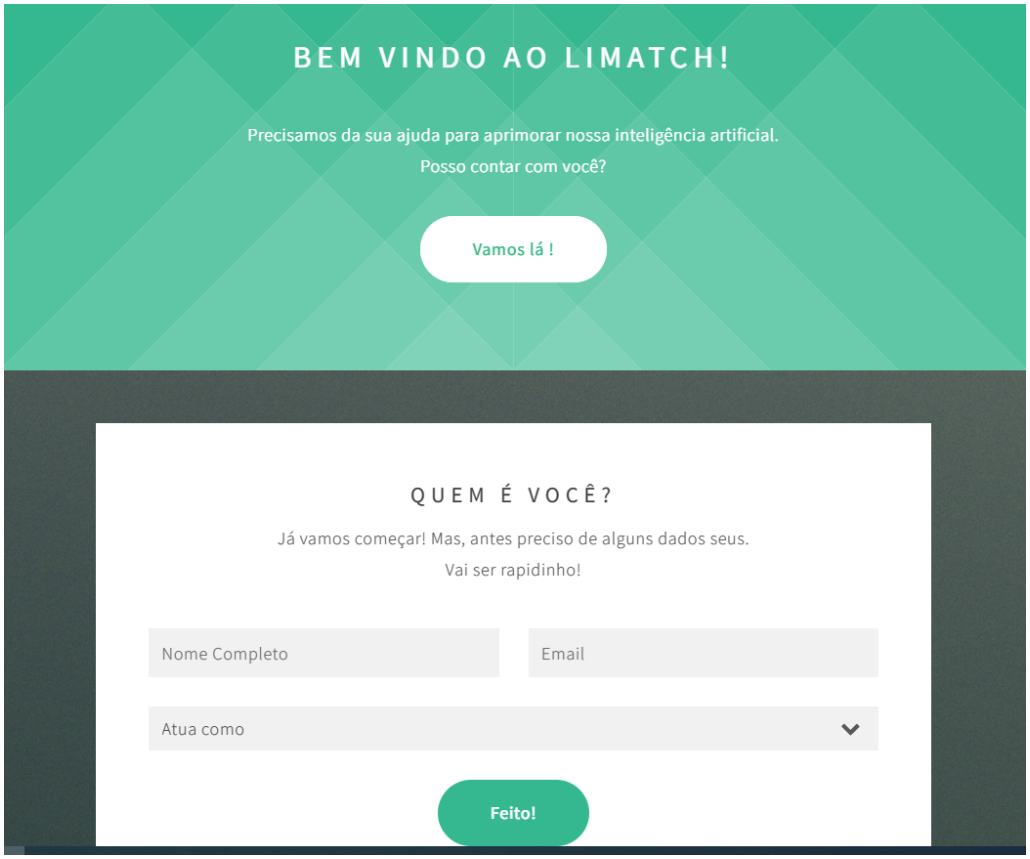


Figure 8 – Screen with register and login fields.

One is for training people about how to identify the ripeness and the quality of lime as can be seen in Figure 9.

And another for actually classifying random lime images, according to Figure 10. The information collected on each screen was persisted in a structured query language database. In this case was used Postgresql as an object-relational database (M. STONE-BRAKER; HIROHAMA, 1990). All the information collected referred to the people who classified limes was saved in a table called "lima_pessoa". This table has 4 fields called "Id", "Nome", "Email" and "Cargo". Id is an auto-increment field, used as the primary key to identify a unique person. "Nome" is a varchar field that contains a person's full name.

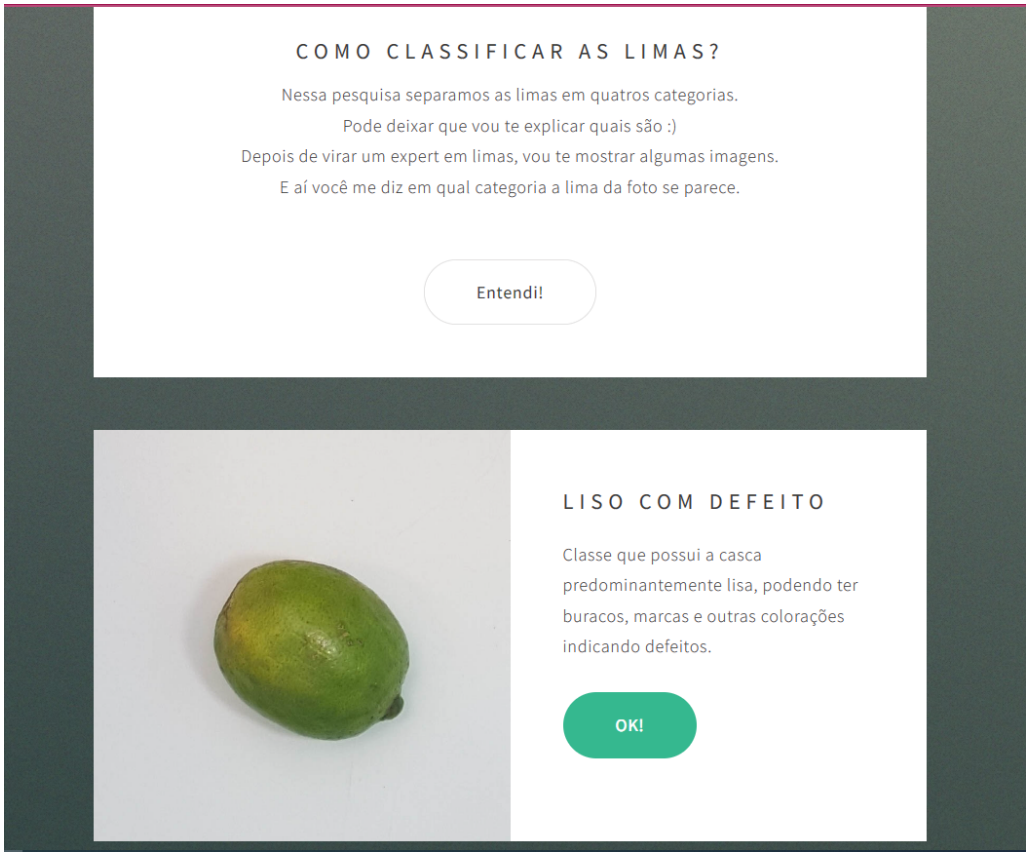


Figure 9 – Screen with different limes classes and their definitions.

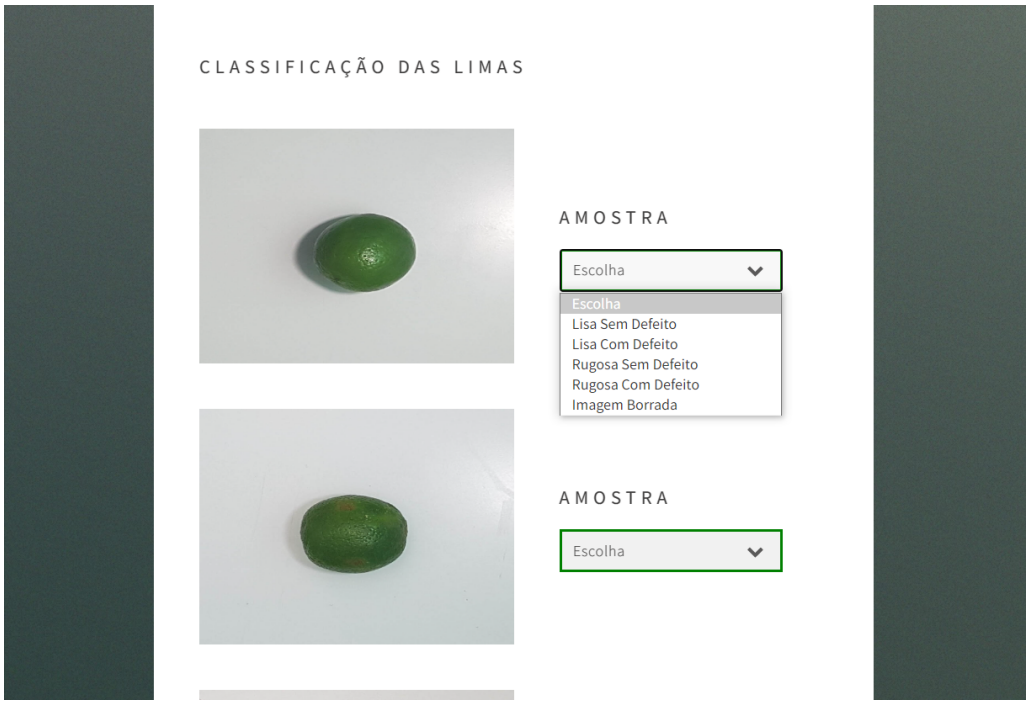


Figure 10 – Screen with different limes classes and their definitions.

“Cargo” is a varchar field that describes if the person is a student, professor, or a person with information technology skills related. “Email” is a varchar field that has people’s email and it must be unique because this field is used to check if the person already classified some limes previously preventing LiMatch to show limes’ image that this person has already done.

Another very important table is called “lima_amostra”. This table has information about each lime image and the lime itself. There are several columns in this table, they could be separated into 2 groups. The first group is about the lime characteristics extracted from the image. These info were input into this table through a Python script and it is not displayed on LiMatch. The relevance of having this info was detailed in Chapter 4. Belong to this group the following fields:

- Color,
- Contrast,
- Correlation,
- Dissimilarity,
- Energy,
- ASM,
- Homogeneity,
- Kurtosis,
- Skewness, and
- Radius.

The second group is related to the fields that are directly used by LiMatch. Id is an auto-increment field, used as the primary key to identify a unique sample of a lime image.“Amostra” is a varchar field that contains the filename of the image, “Base” is a varchar field that states if this image was originally used to train the artificial intelligence models or to test their accuracy. “Data_amostra” is a date field used to record when the image was inserted into the database table. “Grupo” is an integer field used to segment data in two groups, so LiMatch only displays samples that are assigned to a specific group. This field was particularly useful in the strategy to uniform the number of classifications

that a specific lime image was classified. “Tipo” is a varchar field that describes if the image was taken or generated through a data augmentation process.

There is a relational table called “lima_label”. This table contains the final result of a classification, in other words, each line of this table relates a person to a sample and the classification that this person attributed to the sample analyzed. Id is an auto-increment field, used as the primary key to identify a unique classification of a lime done by one person. “Data_hora” is a date field that has the time which the classification was done. “Amostra_id” contains the exact same unique id from the table “lima_amostra” that identifies a unique lime image. “Pessoa_id” contains the exact same unique id from the table “lima_pessoa” that identifies a unique lime image. “Maturacao” is a varchar field that describes if the lime in the image classified is ripe or not, according to the person who classified it. “Defeito” is a varchar field that describes if the lime in the image classified contains defects or not, according to the person who classified it.

It is plausible to highlight that “Maturacao” could assume values as L, which means mature, or R not mature. And “Defeito” could assume values as S, which means defectless, or C with defect. Both “Maturacao” and “Defeito” could be assigned an X value and it means that particular sample was could not have been identified as part of one of the previous categories.

After modeling the database, the challenge was to deploy this web application in a public domain so it could reach as many people as possible. To help with this was employed Heroku (**Heroku**), which is a Platform As A Service (Paas) solution. A Paas solution enables developers to build, run, and operate applications entirely in the cloud.

The methodology applied to set the LiMatch label was basically:

1. A person receives a brief explanation about the different classes of limes and how to identify them;

2. The person sees numerous sample images and classify the limes in them through LiMatch;

3. Limatch get this classification converts it to a label and persists it as a vote on the database,

4. A function called `get_limatch_updated_label` is executed manually and it:

   (a) Get the number of votes for each class from PostGres database and group it

per sample using ”Amostra_id”,

(b) Processes data related to votes for maturity and defect, identifies the entries with the highest votes within each group,

(c) Drop samples which X was the highest voted,

(d) Returns a list of samples and their new major-voted label.

5. A CSV file is generated containing this list and is used to train and test new models.

During all the time that LiMatch was available online approximately 200 people made about 27503 annotations at the whole base. With these newer classifications, was feasible to create another label for the dataset, in this process, 36 images were dropped from the dataset for being label as X by the majority. And 156 images have their label changed as it could be seen with more details on Table 8.

As a matter fact it is important to emphasize that from now and on this work has two labels. The original label, y_o, that were manually annoted by a volunteer and LiMatch label, y_l, that were manually annoted by a set of volunteers.

Table 8 – Number of samples per class in LiMatch's new annotation.

Classes	Number of Samples			
	Original Label		LiMatch Label	
	Train	Test	Train	Test
Mature (L)	600	80	637	91
Not Mature (R)	359	44	291	28
Defective (C)	372	62	374	50
Not Defective (S)	587	62	554	69
Total	1918	248	1856	238

5.1- Experimental procedure applied on LiMatch Label

This section is dedicated to apply the experimental procedure done in Section 4.2 on this new label, called LiMatch label. The application of the experimental procedure using LiMatch label generated Experiment 3 which is about the maturity of the limes and Experiment 4 which concerns the existence of defects on the lime. The parameters selected using a stratified K-fold method within a Bayesian search are described on Section 5.1.1.

5.1.1 Parameter optimization using LiMatch Label

To infer the accuracy of the classification system concerning the maturity and defects of the limes Experiments 3 and 4, respectively, were made using LiMatch label.

Since a new annotation was generated by LiMatch system, it was necessary to look for the best hyperparameters again, with a stratified K-fold method that was applied during Bayesian search and select the best sets . Tables 9, 10 and 11 presented the new hyperparameters chosen for ANN, RF and XGB models respectively.

Table 9 – Bayesian search performed in the parameter space of the training set using ANN on the Original and LiMatch Label. The last four columns are the best parameters for each experiment, with respective label.

Parameter	**Bayesian Search**	**Original Label**		**LiMatch Label**	
		Exp 1	**Exp 2**	**Exp 3**	**Exp 4**
Neurons in hidden layer	2,3, $\cdots$, 500	500	500	500	500
Activation function	('tanh', 'relu')	relu	relu	tanh	tanh
Optimization algorithm	('sgd', 'adam','lbfgs')	adam	adam	adam	adam
ℓ_2 penalty (regularization term α)	1^{-6} ,1^{-5}, 0.0001, 0.05, 1	1^{-6}	1	1	1^{-6}
Learning Rate	('constant', 'adaptive')	constant	adaptive	adaptive	constant
Random State	1, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80,90,100	100	1	90	1

Table 10 – Bayesian search performed in the parameter space of the training set using RF on the Original and LiMatch Label. The last four columns are the best parameters for each experiment with respective label.

Parameter	**Bayesian Search**	**Original Label**		**LiMatch Label**	
		Exp 1	**Exp 2**	**Exp 3**	**Exp 4**
Number of trees	20, 30, $\cdots$, 2000	406	1560	1560	240
Maximum number of features at every split	auto, sqrt , log	sqrt	auto	sqrt	sqrt
Maximum tree depth	10, 11, $\cdots$, 30	18	22	22	18
Minimum number of samples to split	2, 5, 10, 15	2	2	2	2
Minimum number of samples at each node leaf	1, 2, 4, 6	2	1	2	2
Bootstrap	true, false	false	false	false	false

Table 11 – Bayesian search performed in the parameter space of the training set using XGB on the Original and LiMatch Label. The last four columns are the best parameters for each experiment with respective label.

Parameter	**Bayesian Search**	**Original Label**		**LiMatch Label**	
		Exp 1	**Exp 2**	**Exp 3**	**Exp 4**
Number of trees	$10, \cdots, 200$	153	29	159	197
Max Depth	$1, \cdots, 50$	15	15	33	26
Subsample	$0.5, \cdots, 1$	0.85	0.91	0.56	0.56
Learning Rate	$0.01, \cdots, 0.3$	0.3	0.23	0.16	0.19
Colsample by tree	$0.5, \cdots, 1$	0.77	0.82	0.53	0.51

5.2- Results and discussion using LiMatch label

This section is reserved to present the results of the models tuned with the best parameters presented in the previous section. In Section 5.2.1 was discussed the performance of K-fold applied on new experiments. In Section 5.2.2 was presented the results of the experiments 3 and 4 a direct comparison between the results with and without the annotation obtained using LiMatch system.

5.2.1 Results on training and validation sets

The parameter optimization generate new models with a slightly higher accuracy if compared with previous experiments using the Original label as can be seen in Table 12. Despite the increase of accuracy, a minor increase of standard deviation could be perceived if compared experiments 1 and 3, as well, 2 and 4 for each classifier. There are some cases that the standard deviation presented a small reduction or even was stable when compared with results obtained using Original Label. These variations were not great enough and it is not expected a significantly change of the results that will be stated.

Table 12 – Performance of classifiers using K-fold with Original label and LiMatch label.

Classifier	Experiment	Label	Accuracy ρ [%]
Random Forest	Exp 1	Original	83.86 ± 1.99
	Exp 2	Original	82.81 ± 1.87
	Exp 3	LiMatch	85.56 ± 1.56
	Exp 4	LiMatch	86.43 ± 3.27
ANN	Exp 1	Original	68.75 ± 2.97
	Exp 2	Original	72.77 ± 2.39
	Exp 3	LiMatch	69.02 ± 2.25
	Exp 4	LiMatch	78.99 ± 3.46
XGB	Exp 1	Original	83.48 ± 1.82
	Exp 2	Original	82.08 ± 1.61
	Exp 3	LiMatch	84.47 ± 1.86
	Exp 4	LiMatch	86.17 ± 3.02

5.2.2 Results on test dataset

Table 13 presents the performance for the maturity classification related to Experiment 3. The classes mature and immature limes have their values changed from previous annotation. On LiMatch annotation there are 91 samples labeled as mature and 28 labeled as not mature, as was displayed in Table 8. The classifiers XGB and RF achieved 87.39% of hit rate on the test set and the ANN classifier underperformed with 74.79%. An increased performance was noted when compared the results using the original label with the results with LiMatch label. With LiMatch label was possible to increase almost 10% in ANN accuracy, an increase of around 4% was obtained in RF accuracy and for XGB a 7% higher accuracy was achieved . Despite the significant enhancement in performance of the ANN classifier, the RF and XGB classifiers were equivalent and outperforms it

Table 14 presents the performance for the defective classification related to Experiment 4. The classes defective and not defective limes have their values changed from previous label. On LiMatch label there are 50 samples labeled as defective and 69 labeled as not mature, as was displayed in Table 8. With LiMatch label, these classes become imbalanced compared with the previous label that set 62 samples for each class. The classifiers ANN and RF achieved 84.03%, and 92.43% of hit rate on the test set, respectively, showing an expressive increase in performance if compared to results displayed at Table 7. With LiMatch annotation was possible to increase more than 23% in ANN accuracy, an

Table 13 – Experiment 3: Mature $\times$ immature limes classification with LiMatch label and comparison with Original label.

Classifier	Label	T_P	F_P	T_N	F_N	ρ [%]
ANN	Original	57	23	21	23	64.52
	LiMatch	84	7	23	5	74.79
RF	Original	74	6	15	29	83.06
	LiMatch	85	6	9	19	87.39
XGB	Original	69	11	14	30	79.83
	LiMatch	83	8	7	21	87.39

increase of around 13,4% was obtained in RF accuracy and a increase of 10,6% with XGB classifier. Despite the significant enhancement in performance of the ANN classifier, the RF classifier still outperforms it.

Table 14 – Experiment 4: With $\times$ without defects limes classification with LiMatch label and comparison with Original label.

Classifier	Label	T_P	F_P	T_N	F_N	ρ [%]
ANN	Original	17	45	4	58	60.48
	LiMatch	36	14	5	64	84.03
RF	Original	43	19	7	55	79.03
	LiMatch	43	7	2	67	92.43
XGB	Original	38	24	8	54	74.19
	LiMatch	36	14	4	65	84.87

Even though at the Experiment 3 the classifiers RF and XGB have the same performance, at the Experiment 4 the classifier RF was absolute having more than 7% of advantage from XGB classifier. Based on this it is plausible to affirm that RF is the most qualified classifier to evaluate limes using LiMatch label. As could be inferred by Tables 13 and 14 LiMatch did increase the performance results for each problem addressed at the experiments. Further enhancements will be applied using this LiMatch label in order to try to accomplish better accuracies.

5.3- Preprocessing and Feature space mapping using a convolutional neural network

At the Section 5.3.1 we will be doing a brief overview about the deep convolutional neural network architecture and how it was employed at the subsequent sections. The parameter optimization results for each classifier that will be combined with ResNet-152 are presented at Section 5.3.2. In Section 5.3.3 we will be evaluating the training process applied on LiMatch labeled lime dataset ResNet-152 combined with the previous classifiers evaluated in this work. Section 5.3.4 refers to the accuracies found for the applicattion of ResNet-152 by itself, as well, the accuracies of it combined with XGB, ANN and RF.

5.3.1 Deep convolutional neural network: ResNet-152

ResNet-152 is a deep convolutional neural network that is part of the Residual Network (RESNET) family, which introduced the concept of residual learning through skip connections. This architecture significantly improves the training of very deep networks by addressing the vanishing gradient problem, allowing for more efficient training and better performance on complex tasks such as image classification. The ResNet-152, in particular, consists of 152 layers, utilizing a structure known as bottleneck blocks. Each block typically includes three convolutional layers, and the architecture employs skip connections that bypass one or more layers to facilitate gradient flow during backpropagation (HE et al., 2016).

Pre-trained ResNet-152 models are available in popular deep learning frameworks, such as PyTorch and TensorFlow, with weights trained on the ImageNet dataset. ImageNet is a large database of images, containing millions of labeled pictures grouped into thousands of categories. It is widely used for training and testing machine learning models, especially for tasks like image classification and object detection (DENG et al., 2009). These pre-trained models can be adapted or fine-tuned to specific tasks, making them highly flexible for transfer learning applications. This ability to adjust the model for different purposes makes them useful in a wide range of areas (CHOLLET, 2017).

To address the task of lime classification, we employed a deep learning model based on the ResNet-152 architecture. To leverage transfer learning and accelerate convergence, the model was initialized with pre-trained weights from the ImageNet dataset (DENG et al., 2009).

The original fully connected layers of ResNet-152 were excluded (using the function parameter include_top=False) and replaced with a custom classification head specifically designed for binary classification. A global average pooling layer (function parameter pooling='avg') was used after the convolutional base to reduce the spatial dimensions, followed by a dense layer with 1024 units and Relu activation to capture high-level features. The final layer consisted of a single neuron with a sigmoid activation function, enabling binary classification. The model was compiled using the Adam optimizer for efficient parameter updates, binary crossentropy as the loss function to handle the binary target, and accuracy as the evaluation metric. All these parameters were used from the builtin functions available at TensorFlow and Keras Python library (ABADI et al., 2015).

The input used for ResNet-152 was the train dataset with the LiMatch label with no previous image processing. It is relevant to emphasize that for all ResNet-152 results the image pre-processing and the feature space mapping handcrafted described on Section 4.1 were not used. The intention here is to compare and analyze which method would be more accurate, which will be clarified at the Section 5.3.4.

5.3.2 Parameter optimization for classifiers combined with ResNet-152

This section is reserved to describe the new configuration found by the Bayesian search routine for the XGB, ANN and RF classifiers when employed using a ResNet-152 model as feature extractor. It is important to emphasize that the same process applied previously in order to found the best parameters for these classifiers was replicated here (K-fold with Bayesian search).

To infer the accuracy of the classification using a combined evaluation (using ResNet-152 combined with other classifiers) two experiments were performed. The Experiment 5 was designed to measure accuracies from combined models and also an end to end ResNet-152 to evaluate the classification of limes when it comes to maturity,

and Experiment 6 has the same objective except that it was designed to classify limes regards the presence of defects or not, both were made using LiMatch label.

Table 15, Table 16 and Table 17 presented the new hyperparameters chosen for ResNet-152 + ANN, ResNet-152 + RF and ResNet-152 + XGB models respectively.

Table 15 – Best parameters of the training set using ResNet-152 + ANN with LiMatch label detailed for each experiment.

Parameter	**Bayesian Search**	**Original Label**	
		Exp 5	**Exp 6**
Neurons in hidden layer	$2, \cdots, 500$	500	2
Activation function	('tanh', 'relu')	relu	tanh
Optimization algorithm	('sgd', 'adam','lbfgs')	lbfgs	lbfgs
ℓ_2 penalty (regularization term α)	1^{-6} ,1^{-5}, 0.0001, 0.05, 1	0.05	0.0001
Learning Rate	('constant', 'adaptive')	adaptive	adaptive
Random State	1, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80,90,100	5	100

Table 16 – Best parameters of the training set using ResNet-152 + RF with LiMatch label detailed for each experiment.

Parameter	**Bayesian Search**	**Original Label**	
		Exp 5	**Exp 6**
Number of trees	$20, 30, \cdots, 2000$	680	1340
Maximum number of features at every split	auto, sqrt , log	sqrt	sqrt
Maximum tree depth	$10, 11, \cdots, 30$	30	12
Minimum number of samples to split	$2, 5, 10, 15$	15	2
Minimum number of samples at each node leaf	$1, 2, 4, 6$	6	1
Bootstrap	true, false	true	false

5.3.3 Results on training and validation sets using ResNet-152

Along the training of the ResNet-152 model the results of losses and accuracies were saved for analyzing the convergence and efficiency of the model along the epochs.

Table 17 – Best parameters of the training set using ResNet-152 + XGB with LiMatch label detailed for each experiment.

Parameter	**Bayesian Search**	**Original Label**	
		Exp 5	**Exp 6**
Number of Trees	$10, \cdots, 200$	51	166
Max Depth	$1, \cdots, 50$	16	1
Subsample	$0.5, \cdots, 1$	0.78	0.99
Learning Rate	$0.01, \cdots, 0.3$	0.08	0.25
Colsample by tree	$0.5, \cdots, 1$	0.82	0.67

For the training and validation stage, 75 epochs were defined with a early stop configured for a 25 patience, this patience parameters refers to the number of epochs with no improvement after which training will be stopped.

The loss values indicate whether the model is effectively minimizing the error between predictions and actual labels during training. Accuracy provides a direct measure of the model's ability to make correct predictions on the training and validation sets.

Figure 11 display the curves related to validation and training process for accuracy and losses for the Experiment 5 which is related to maturity presented by limes. Losses curves were normalized using (23), where $Loss_{\min}$ was the minimum value obtained and $Loss_{\max}$ was the maximum value obtained along the training or the validation process. The normalization was performed to ensure comparability and better visualization.

$$\text{Normalized Loss} = \frac{\text{Loss} - \text{Loss}_{\text{min}}}{\text{Loss}_{\text{max}} - \text{Loss}_{\text{min}}} \tag{23}$$

Figure 11 illustrates the training loss having a sharp drop during the initial epochs and then leveled off near zero, showing the model's ability to optimize for the training data. The validation loss curve suffer a huge variation but in the first iteration but it remains stable next to zero, which is a sign that this model may have a good capability of generalization. At the same figure it is possible to observe that the training and validation accuracies consistently increased over the epochs stabilizing at a high level, 96% at later epochs. For validation accuracy curve a higher variability specially when it comes for the initial epochs, but ultimately converged to a relatively stable trend above 85%.

Figure 12 displays the curves related to validation and training process for accuracy and losses for the Experiment 6 which is related to the existence of defects on limes. Very similar to Figure 11, at Figure 12 it is observed a training loss curve with a convergence

tending to zero, it relevant to comment that for this Experiment we took less epochs to achieve the same level achieved by the model using ResNet-152 employed for the Experiment 5. The validation loss curve has the same behavior, relatively low, which lead us to comprehend that the model may be able to perform a good generalization.

As shown at the Figure 12 we have the training and validation accuracies curves. Referring to the training and validation accuracy behave, it seems plausible to assure that despite of the variance performance both end up oscillating around a 98%, for training curve and 84% for validation curve.

The results demonstrate that along the training and validation phase ResNet-152 models effectively captured the complex patterns in the data while maintaining robust generalization to unseen validation data. However, the observed variability in validation accuracy and spikes in validation loss suggest areas for further investigation.

As mentioned in Section 5.3.2, the classifiers XGB, ANN and RF were trained once more for each experiment, but this time with the features provided by ResNet-152. The results of the training process for each combination are listed at Table 18. From Table 18 which is related to Experiment 5, could be perceived a strong accuracy of ResNet-152 + RF while using LiMatch label, highlighting a good effectiveness at the training process when combining ResNet-152's deep feature extraction with Random Forest's classification. At the same experiment, ResNet-152 + XGB reached 85.20 ± 1.69 of accuracy which is tecnically tied with ResNet-152 + RF that presents a 85.80 ± 1.54 accuracy, if we consider the standard deviation.

The ResNet-152 + ANN combination, however, is comparatively less effective, achieving 79.21 ± 2.92, which suggests that ANN might not fully exploit ResNet-152's features.

Concerning to Experiment 6, the results shown at Table 18 inferred a significant drop in performance that is observed across all classifiers, possibly indicating challenges in generalization. ResNet-152 + Random Forest and ResNet-152 + XGB both show reduced accuracy in Exp 6, with 63.07 ± 2.20 and 65.20 ± 2.67, respectively. ResNet-152 + ANN has the lowest accuracy in Experiment 6 at 65.55 ± 2.51, further indicating that this architecture may not be as effective with deep feature from ResNet-152 at the the test step.

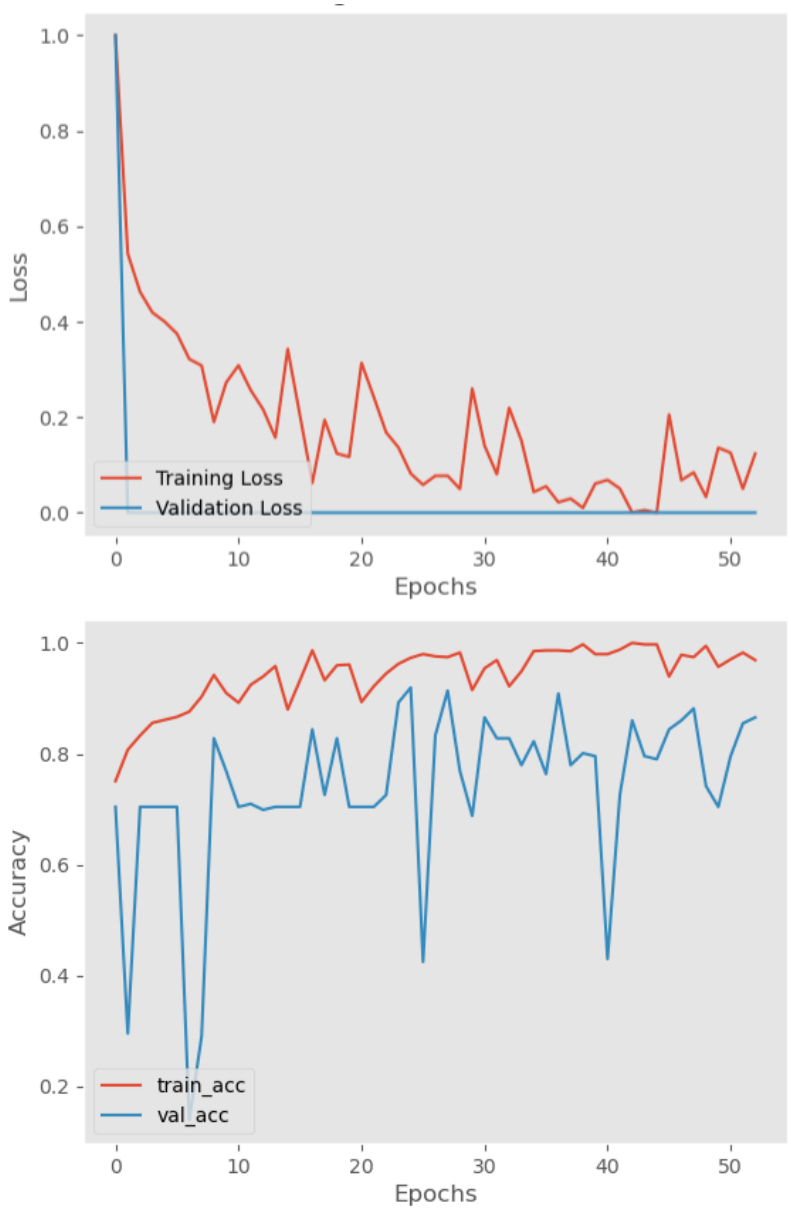


Figure 11 – Loss and Accuracy, respectively, along the epochs training and validation process for Experiment 5 related with maturity classification.

Table 18 – Performance of classifiers combined with ResNet-152 using K-fold.

Classifier	Experiment	Label	Accuracy ρ [%]
ResNet-152 + ANN	Exp 5	LiMatch	79.21 ± 2.92
	Exp 6	LiMatch	65.55 ± 2.51
ResNet-152 + RF	Exp 5	LiMatch	85.80 ± 1.54
	Exp 6	LiMatch	63.07 ± 2.20
ResNet-152 + XGB	Exp 5	LiMatch	85.20 ± 1.69
	Exp 6	LiMatch	65.20 ± 2.67

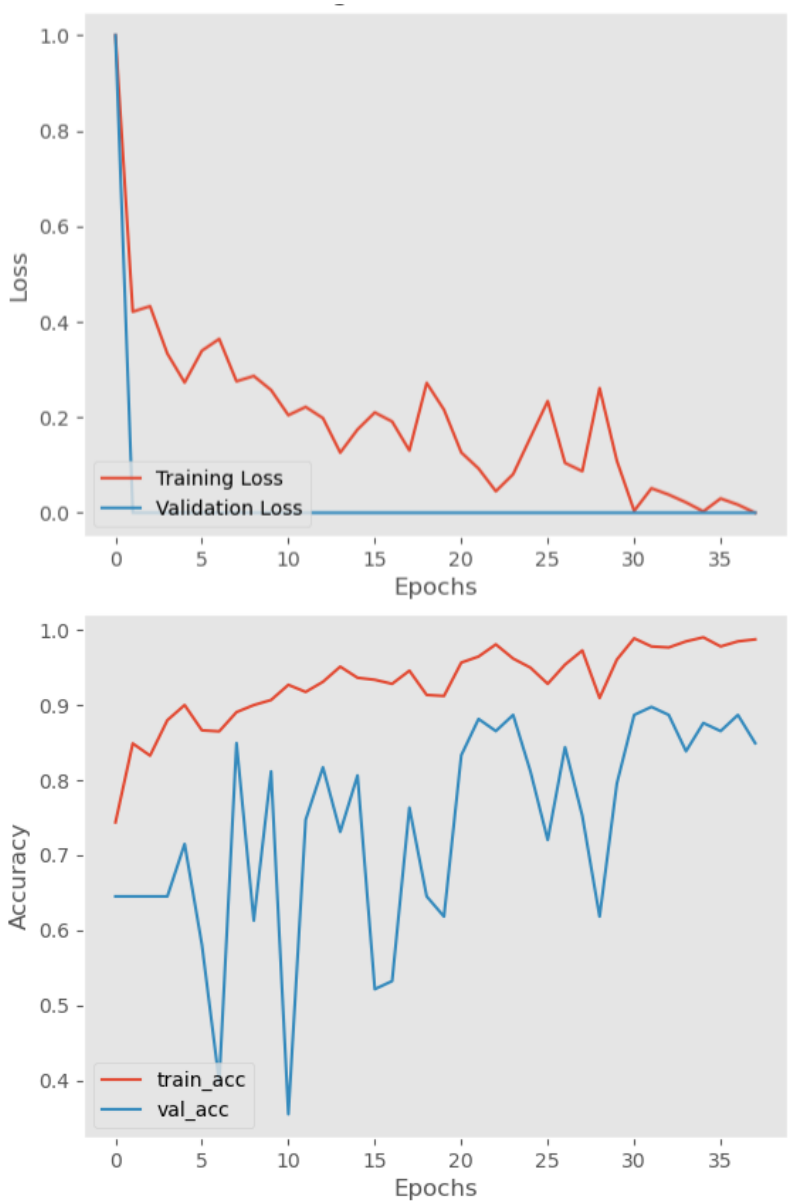


Figure 12 – Loss and Accuracy , respectively, along the epochs training and validation process for Experiment 6 related with the presence of defects classification.

5.3.4 Results on test dataset using ResNet-152

In Experiment 5, we observe the classification performance for distinguishing between mature and immature limes with the LiMatch label. The baseline ResNet-152 model achieves the highest accuracy 85.75% indicating a good performance in correctly classifying limes. The combination of ResNet-152 with XGB almost reach the same result of accuracy achieving 84.03% presenting a the same distribution of true and false positives,

indicating that this pairing has a comparable but slightly lower effectiveness compared to the base ResNet-152 model alone. On the other hand, combined ResNet-152 with ANN and ResNet-152 with RF have the same performance, not very far from the other ones. This experiment results suggests that the combination of the ResNet-152 with other classifiers do not significantly enhance performance for the identification of mature or immature limes. These accuracies were presented at Table 19.

Table 19 – Experiment 5: Mature $\times$ immature limes classification with LiMatch label using ResNet-152.

Method	Label	T_P	F_P	T_N	F_N	ρ [%]
ResNet-152	LiMatch	84	7	10	18	85.75
ResNet-152 + ANN	LiMatch	82	9	11	17	83.19
ResNet-152 + RF	LiMatch	81	10	10	18	83.19
ResNet-152 + XGB	LiMatch	84	7	12	16	84.03

Table 20 provides the hit rate results on test dataset using LiMatch label for Experiment 6. In Experiment 6, focused on differentiating limes with and without defects, the ResNet-152 model alone again achieves the highest accuracy, reaching 90.86% which reinforces ResNet-152's strong capability to generalize across the test set in this defect classification task. However, when combined with other classifiers, it is evident that a significant drop in performance occurs. ResNet-152 + ANN achieves 68.90% accuracy, while ResNet-152 + RF reaches 71.43%, both with higher false positive and false negative counts, likely indicating overfitting or difficulty in transferring learned features across combined model structures. About the combination of ResNet-152 + XGB may be understood it generated the lowest performance with an accurracy of 57.98% which could be a sign of challenges in reconciling the feature extraction capabilities of ResNet-152 with XGB's boosting mechanism.

An insight brought from both experiments is that the combined models resulted in an underscore ResNet-152's effectiveness as a standalone model, which could be an indication that introducing ensemble methods does not necessarily improve, and may even prejudice, performance for these particular tasks.

The results achieved with ResNet-152 standalone were the best compared with the others experimented within the Experiment 5 and 6. However, the results achieved by it did not exceed the results obtained with Experiment 3 and 4. This matter will be

Table 20 – Experiment 6: With $\times$ without defects limes classification with LiMatch label using ResNet-152.

Method	Label	T_P	F_P	T_N	F_N	$\rho\,[\%]$
ResNet-152	LiMatch	42	8	3	66	90.86
ResNet-152 + ANN	LiMatch	22	28	9	60	68.90
ResNet-152 + RF	LiMatch	27	23	11	58	71.43
ResNet-152 + XGB	LiMatch	14	36	14	55	57.98

discussed with proper depth at Chapter 7.

6- Data augmentation

6.1- Data Augmentation

In the fields of computer vision and artificial intelligence, imbalanced datasets are a significant challenge for machine learning models. An imbalanced dataset occurs when one class (the minority class) has significantly fewer instances than the other (the majority class). This imbalance can lead to biased models that perform poorly on the minority class (HE; GARCIA, 2009).

As was detailed on Chapter 5, more specifically at the Table 8 we must consider the dataset created for this work as an imbalanced dataset, and after doing the relabeling process with LiMatch it still remained imbalanced. Therefore, addressing class imbalance is a crucial step in improving model generalization and overall performance.

One of the most effective techniques to address this issue is the Synthetic Minority Oversampling Technique (SMOTE), presented by Chawla (CHAWLA et al., 2002). The methodology presented by this study does not merely replicates minority class samples, with SMOTE technique synthetic instances are generated by between the minority class instances that were representing this class.

At this chapter the main concepts of SMOTE and their variants will be explained in Sections 6.1.1, 6.1.2 and 6.1.3. Subsequently the results and discussions about using data augmentation will be detailed in Section 6.2

6.1.1 SMOTE

SMOTE is a statistical method used generally to minimize class imbalance in datasets. It interpolates creating new samples between existing minority class examples and the nearest neighbors along the lines connecting them in feature space (CHAWLA et al., 2002). This synthetic data generation process creates new examples that add to

the training data, and helps the model to get more examples of this class without the risk of having the same sample leading to a possible overfitted model.

The SMOTE technique do the following procedure, first it identifies the minority class, then for each instance in the minority class, the algorithm select K nearest neighbors within the minority class. After that a random neighbor from the K nearest neighbors is selected. Then between the pairs of points generated by the sample and each of its neighbors, a new synthetic data is generated by taking linear combinations of the features of the selected instances and their neighbors (CHAWLA et al., 2002). This process is named as interpolation and formula is defined at Equation (24).

$$x_{\text{new}} = x_i + \lambda \cdot (x_k - x_i) \tag{24}$$

where:

- x_{new} is the synthetic sample generated,
- x_i is a minority class sample,
- x_k is a randomly selected minority class neighbor of x_i,
- λ is a random value in the range $[0, 1]$.

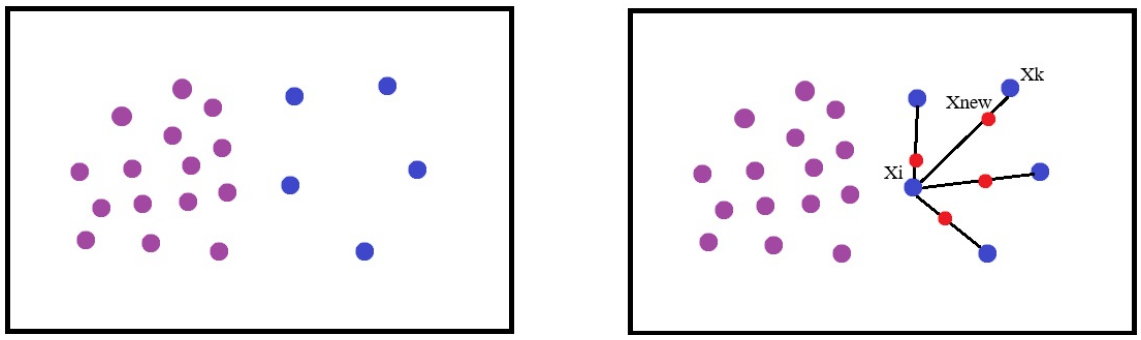


Figure 13 – On the left, a representation of a dataset with two classes. On the right, a representation of interpolation method performed by SMOTE technique.

Figure 13 on the left, presents a representation of a dataset with two classes (violet and blue), on the right it is presented how the interpolation used in SMOTE works at the minority class in order to artificially generate new samples x_{new} here shown as red dots. SMOTE might be very useful to create artificial samples and it helps to balance classes. However the synthetic data generated is linearly dependent, which can cause a bias and consequently produce an overfitted model. That was the reason why alternatives based

on SMOTE have been developed that focus on improving the limitations of the original SMOTE technique.

6.1.2 Borderline SMOTE

In 2005, Hui Han introduced a new variation of SMOTE. At his paper, he proposes improvements to the popular SMOTE algorithm to address the challenges of imbalanced datasets. There was introduced two variants, which focus specifically on minority class samples near the boundary between classes, where classification errors are most likely to occur (HAN; WANG; MAO, 2005).

Unlike traditional SMOTE, which applies oversampling across the entire minority class, Bordeline SMOTE categorize minority class instances into three groups: safe, danger, and noise according the distribution of minority and majority class neighbors around each minority class sample and then prioritize instances labeled as danger by their proximity to majority-class neighbors, ensuring targeted improvements in classification performance. These approaches are designed to avoid oversampling noisy instances, which could degrade model performance, and instead focus on areas where decision boundaries are ambiguous (HAN; WANG; MAO, 2005) (LEMAÎTRE; NOGUEIRA; ARIDAS, 2017).

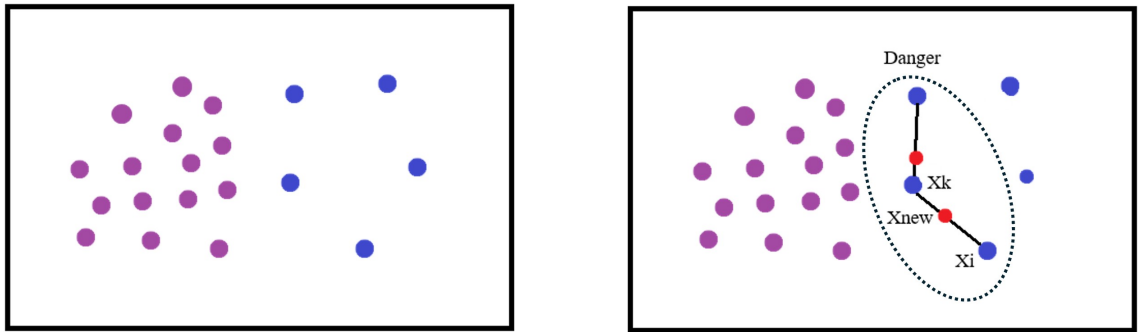


Figure 14 – On the left, a representation of a dataset with two classes. On the right, a representation of the instances segmented as Danger, where the SMOTE is applied, becoming a Borderline Smote.

At Figure 14, it is possible to see instances marked as Danger where the SMOTE technique was applied using the same interpolation Equation (24). The difference here

is that this process only is done with the one marked as Danger which compose the so called bordeline.

6.1.3 ADASYN

Adaptive Synthetic (ADASYN) was create with the intention of address class imbalance in machine learning by generating new synthetic samples. With ADASYN instances are created using the same linear interpolation, presented on Equation 24, between a minority class sample and one of its K-nearest neighbors. A major difference here when compared to SMOTE, is that ADASYN calculates a density distribution for minority class samples having the difficulty of their classification as base. So, more synthetic samples are generated for instances surrounded predominantly by majority class neighbors, which are harder to learn, while fewer samples are generated for easier instances. By this logic it is ensured that the model focus on difficult regions of the feature (HAN; WANG; MAO, 2005).

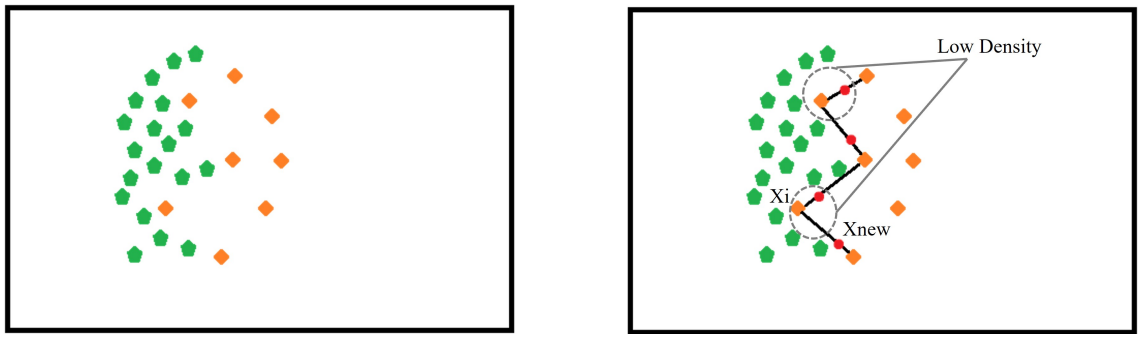


Figure 15 – On the left, a representation of a dataset with two classes. On the right, a representation of the instances of the minority class with orange color and the low density regions demarcated within the grey dashed circle, the samples inside of it where used to generate red dots instances.

Presented at Figure 15, on the left is the representation of a dataset with two classes the green which is the majority class and the orange which is the minority class. On the right is represented how a ADASYN technique would performed when applied to this dataset. One can deduce that the instances inside the low density area were selected to generate new synthetic samples, red dots, using interpolation.

6.2- Results and discussions using Data Augmentation with LiMatch label

After a careful review from the SMOTE variants, it was plausible to infer that SMOTE generates synthetic samples uniformly across the minority class, making it effective for datasets where minority instances are well-distributed and not heavily overlapped by the majority class. However, it may struggle in boundary regions. To address this, Borderline SMOTE focuses on creating samples near decision boundaries, enhancing the model's performance in overlapping regions. ADASYN adapts by generating more synthetic samples for instances that are harder to classify, proving particularly effective in datasets with complex or non-linear decision boundaries (AZHAR et al., 2023).

This section was created,With the intention to analyze and compare the performance of these techniques, at the designed dataset proposed, at Chapter 3, with the LiMatch labeling refinements described at Chapter 5.

This section will introduce two experiments. Both of them were implemented using a Python library which the development was publish at this paper (LEMAÎTRE; NOGUEIRA; ARIDAS, 2017). The structure of Experiments will remain the same. Experiment 7 will test the maturity level of the limes while Experiment 8 will classify limes according to the presence or absence of defects. The SMOTE method will be applied at the features extracted from the dataset using the handcrafted process. Another aspect both experiments have in common is that the SMOTE techniques were used together with the XGB classifier.

The XGB classifier was choosen between RF, ANN and Resnet-152 for having the fastest model train time and also be one of the best in accuracy achieved in previous results. To elucidate XGB took in average 2 minutes to train a model while RF took around 5 minutes, ANN was up to 12 minutes and ResNet-152 was the slowest taking around 20 minutes to be trained.

At Section 6.2.1 will be presented the best parameters for XGB classifiers and the SMOTE variants, further at Section 6.2.2 the results obtained at the training and validation phase will be presented follow by Section 6.2.3 where the results on test dataset will discussed.

6.2.1 Parameter optimization sets for Data Augmentation methos of oversampling classes

For the parameter optimization configuration again a stratified K-fold method was applied during Bayesian search and select the best sets for the XGB classifier combined with a SMOTE method, chosen by its average accuracy. Tables 21, 22 and 23 display best parameters found for each experiment using SMOTE, Borderline SMOTE and ADASYN respectively.

Table 21 – Best parameters of XGB the training set using SMOTE with LiMatch label for oversampled class.

Parameter	**Bayesian Search**	**Oversampled Classes**	
		Exp 7	**Exp 8**
Number of trees	$10, \cdots, 200$	143	111
Max Depth	$1, \cdots, 50$	18	40
Subsample	$0.5, \cdots, 1$	0.55	0.71
Learning Rate	$0.01, \cdots, 0.3$	0.30	0.29
Colsample by tree	$0.5, \cdots, 1$	1	0.50

Table 22 – Best parameters of XGB the training set using Borderline SMOTE with LiMatch label for oversampled classes.

Parameter	**Bayesian Search**	**Oversampled Classes**	
		Exp 7	**Exp 8**
Number of trees	$10, \cdots, 200$	200	150
Max Depth	$1, \cdots, 50$	9	13
Subsample	$0.5, \cdots, 1$	0.66	0.63
Learning Rate	$0.01, \cdots, 0.3$	0.21	0.07
Colsample by tree	$0.5, \cdots, 1$	1	0.91

Table 23 – Best parameters of XGB the training set using ADASYN with LiMatch label for oversampled classes.

Parameter	Bayesian Search	Oversampled Classes	
		Exp 7	Exp 8
Number of trees	$10, \cdots, 200$	200	200
Max Depth	$1, \cdots, 50$	21	50
Subsample	$0.5, \cdots, 1$	0.99	0.5
Learning Rate	$0.01, \cdots, 0.3$	0.07	0.19
Colsample by tree	$0.5, \cdots, 1$	0.60	0.5

6.2.2 Results on training and validation sets with oversampled classes

As stated previously the SMOTE process was applied right after the extraction of features using the handcrafted process described at Chapter 4. This process results in a vector of continuous features and that was the input for the SMOTE techniques which used a sampling strategy equals to 1.5, that means that the desirable ratio between the minority class after resampling and the majority class should be 1.5. This coefficient was proposed after a testing with it equals to 1 and having poor accuracies results.

Table 24 – Number of train samples per class with Data Augmentation methods.

Experiment	Classes	None	SMOTE	Borderline SMOTE	ADASYN
Exp 7	Mature (L)	637	955	955	848
	Not Mature (R)	291	955	955	974
Exp 8	Defective (C)	374	831	831	848
	Not Defective (S)	554	831	831	744

Table 24 presents the results of each class for each experiment with each SMOTE variant and also display the original value at column “None” which could be seen at Table 8. After a comparison, it may be understood that both SMOTE and Borderline SMOTE not only balance the classes by using the ratio times the amount of instances of the majority class, but turned the imbalanced dataset to a balanced one. The same is not true for ADASYN. ADASYN does not aim to create an exact balance between classes but rather increases the representation of the minority class proportionally based onthe density of the minority class. As a result the number of instances of each class may vary, leading to

a slightly unbalanced dataset.

Considering the specific amount of instances for each class and SMOTE method the XGB classifier defined by the best parameters displayed at Section 6.2.1 was trained and validated using train dataset resampled. The average accuracy and their respective standard deviation obtained during the training process for each experiment and Data Augmentation method were stated at Table 25.

As could be perceived at Table 25 all the accuracies were higher than 90%, with SMOTE technique having highest accuracy but also higher standard deviation. In general these accuracies could be a sign of a strong performance ate the train and validation process, but we should be careful to not have a overfitted model.

Table 25 – Performance of XGB using K-Fold with Limatch Label with oversampled class.

Data Augmentation	Experiment	Label	Accuracy ρ [%]
SMOTE	Exp 7	Limatch	93.58 ± 2.99
	Exp 8	Limatch	93.02 ± 2.89
Borderline SMOTE	Exp 7	Limatch	92.87 ± 2.73
	Exp 8	Limatch	91.79 ± 2.20
ADASYN	Exp 7	Limatch	92.50 ± 2.45
	Exp 8	Limatch	91.38 ± 2.80

6.2.3 Results on test set with Data Augmentation oversampled classes

This section is going to to evaluate the results presented using SMOTE, Borderline SMOTE and ADASYN with the previous presented models at Section 6.2.1.

At Table 26 its possible to see that all results were very closed from each other having a slightly leverage for ADASYN which yield 89.92% of accuracy. This result could be interpreted as a good performance from the density minority algorithm which probably helped to determine particular cases that the other SMOTE variants did not got right. While SMOTE and Borderline SMOTE achieved nearly identical results, indicating that the decision boundary refinement provided by Borderline SMOTE had minimal impact in this scenario.

At Table 27 Borderline SMOTE was the most effective, achieving the highest accuracy 94.12% due to its focus on overlapping regions near the decision boundary.

Table 26 – Experiment 7: Mature $\times$ immature limes classification with LiMatch Label and comparison between data augmentation methods.

Data Augmentation	Label	T_P	F_P	T_N	F_N	ρ [%]
SMOTE	LiMatch	83	8	6	22	88.24
Borderline SMOTE	LiMatch	83	8	5	23	89.08
ADASYN	LiMatch	85	6	6	22	89.92

SMOTE and ADASYN, although helpful, achieved lower accuracies 89.92% and 88.24%, respectively, indicating that their synthetic sample generation was less effective in capturing the nuanced distinctions in defective and non defective lime classifications.

Table 27 – Experiment 8: With $\times$ without defects limes classification with LiMatch Label and comparison between data augmentation methods.

Data Augmentation	Label	T_P	F_P	T_N	F_N	ρ [%]
SMOTE	LiMatch	41	9	3	66	89.92
Borderline SMOTE	LiMatch	46	4	3	66	94.12
ADASYN	LiMatch	39	11	3	66	88.24

7- Concluding remarks and future work

This work describes a Tahiti lime classification using computer vision and machine learning. It classifies limes according to the maturity and presence/absence of defects employing classifiers such as : XGB, ANN, RF and also a deep convolutional neural network ResNet-152. Another topic of this work was a data augmentation study and the use of SMOTE to oversampled the training dataset. It was proposed a handcraft method that performs classification in a feature space that combines shape, color, and texture measures. Also, we produced a dataset composed of images from hundreds of limes in several conditions.

In order to reduce the bias of the annotation done for only one person, here referred as Original, a system called LiMatch was proposed to acquire new sources for relabeling the dataset and generate a new label for based on the opinion of multiple people.

LiMatch did happen to be a better label process so it was used in the other half of experiments. Subsequently, it was experienced if a ResNet-152 would work good as a feature extractor combined with the precious classifiers proposed or even if a ResNet-152 standalone would outperform the higher hit rate for each experiment. After the experiments, the methods that used ResNet-152 did not achieve best results in terms of accuracy when compared to ones designed in this work, here called handcraft method. At this point, we end up with a highly imbalanced dataset which could harm the performance, so data augmentation methods, more specifically SMOTE, Borderline SMOTE and ADASYN, were with handcrafted feature extraction, LiMatch Label and XGB, these were selected for being the best set in terms of accuracy.

Multiples experiments were conducted in order to infer which is the best solution for the issue of classifying Tahiti limes. Figure 16 was created to elucidated how the experiments were organized and how compared the results between them.

There were three major groups of experiments:

- Relabel: Related to the original label process and enhancement with LiMatch.
- ResNet-152: Related to the use of deep convolutional neural networks combined with other classifiers or used as a standalone solution.

- Data Augmentation: Related to the use of SMOTE and its variants methods combined with XGB classifier and LiMatch label.

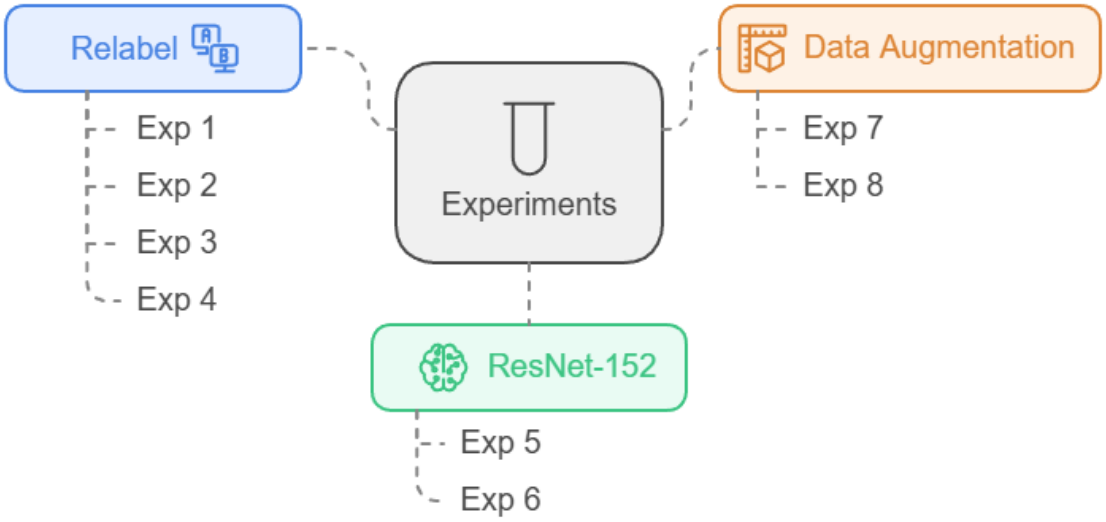


Figure 16 – Organization of the experiments along this work.

Inside each major group of experiments there are odd experiments that address the same type of classifiers related to maturity of the lime and even experiments which concern the presence or absence of defects in a Tahiti Lime. Table 28 and Table 29 display the best accuracy results for each type of experiment.

Table 28 – Mature $\times$ immature limes classification best results.

Experiment	Method	Label	T_P	F_P	T_N	F_N	ρ [%]
7	Handcrafted + XGB + ADASYN	LiMatch	85	6	6	22	89.92
3	Handcrafted + RF	LiMatch	85	6	9	19	87.39
3	Handcrafted + XGB	LiMatch	83	7	7	21	87.39
5	ResNet-152	LiMatch	84	7	10	18	85.75

At Table 28 in Experiment 3, RF demonstrated superior performance with an accuracy 87.39% under the LiMatch label. This out performance highlights RF's robustness in leveraging diverse feature spaces effectively, aligning with findings that tree-based ensemble models often excel in structured classification tasks. However, XGB had almost the same performance in aspects of true positive counters, and it score just the same in terms of accuracy.

Experiment 5 shows that ResNet-152 had the lower accuracy between all the

best performers from each group of experiments, althought it was not far from the other compositions. That suggests ResNet-152's inherent feature extraction capability remains strong, even without supplementary classifiers, as reported in recent studies emphasizing ResNet-152's architecture's depth as a key factor.

Experiment 7 demonstrated that augmentation methods like Borderline-SMOTE and ADASYN improved classification outcomes. Borderline-SMOTE achieved a higher accuracy of 89.08% showing its efficacy in addressing boundary regions. If compared to the same method without augmentation operation we could notice a loss of 2.5% percent in accuracy.

Table 29 – With $\times$ without defects limes classification best results.

Experiment	Method	Label	T_P	F_P	T_N	F_N	ρ [%]
8	Handcrafted + XGB + Borderline	LiMatch	46	4	3	66	94.12
4	Handcrafted + RF	LiMatch	43	7	2	67	92.43
6	ResNet-152	LiMatch	42	8	3	66	90.86

At Table 29 in Experiment 4, RF was again one of the best-performing classifiers, achieving 92.43% accuracy. At Experiment 6, ResNet-152 alone achieved 90.86% accuracy, while combining it with ANN and XGB resulted in lower accuracies. This decline suggests that complex neural architectures like ResNet may not always benefit from supplementary classifiers, especially when misaligned with the primary architecture's feature extraction process. In Experiment 8, Borderline-SMOTE was clearly the most effective method with 94.12%. The prioritization of decision boundaries in Borderline-SMOTE likely contributed to its success.

Across all odd and even experiments, RF and XGB demonstrated consistent superiority among classifiers. This aligns with its established reputation in machine learning for high accuracy and interpretability . While ANN showed improvements with LiMatch labels, its performance lagged behind other classifiers, suggesting its limitations in capturing subtle feature distinctions within not large datasets.

Data augmentation techniques significantly enhanced classification results. Borderline-SMOTE's focus on critical regions close to the decision boundary proved particularly effective in both experiment scenarios. ADASYN's adaptive strategy was less consistent, highlighting the trade-off between sample generation diversity and model stability in imbalanced dataset

7.1- Future work

This research highlights several promising ways for further exploration in lime classification using data augmentation and machine learning techniques:

- Advanced Augmentation Strategies: Investigate hybrid augmentation methods (HE; GARCIA, 2009)
- Deep Learning Architectures: Explore state-of-the-art transformer-based models or graph neural networks to improve feature extraction and decision boundary accuracy (HAN; WU; YANG, 2022).

References

ABADI, Martín et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. Google Brain: Google Brain, 2015. Software available from tensorflow.org.

ACHARYA, Tinku; RAY, Ajoy K. **Image processing: principles and applications**. Hoboken, NJ, USA: John Wiley & Sons, 2005.

AZHAR, Nur Athirah et al. An Investigation of SMOTE Based Methods for Imbalanced Datasets With Data Complexity Analysis. **IEEE Transactions on Knowledge and Data Engineering**, v. 35, n. 7, p. 6651–6672, 2023.

BALLARD, Dana Harry; BROWN, Christopher M. **Computer Vision**. 1st. Englewood Cliffs, NJ, USA: Prentice Hall Professional Technical Reference, 1982.

BLASCO, J. et al. Recognition and classification of external skin damage in citrus fruits using multispectral data and morphological features. **Biosystems Engineering**, v. 103, n. 2, p. 137–145, 2009.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BREIMAN, Leo. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001.

CARUANA, Rich et al. An empirical comparison of supervised learning algorithms. **Proceedings of the International Conference on Machine Learning (ICML)**, 2008.

CHAWLA, Nitesh V et al. SMOTE: Synthetic Minority Over-sampling Technique. **Journal of Artificial Intelligence Research**, v. 16, p. 321–357, 2002.

CHEN, Kexin. Research on Popular Machine Learning Algorithms. In: IEEE. 2023 IEEE 6th International Conference on Information Systems and Computer Aided Education (ICISCAE). Dalian, China: IEEE, 2023. P. 14–21.

CHEN, Tianqi. **XGBoost Documentation**. Online, 2023. Accessed: 2023-01-20.

CHEN, Tianqi; GUESTRIN, Carlos. XGBoost: A Scalable Tree Boosting System. In: ACM. PROCEEDINGS of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, CA, USA: ACM, 2016. P. 785–794.

CHERNOV, Vladimir; ALANDER, Jarmo; BOCHKO, Vladimir. Integer-based accurate conversion between RGB and HSV color spaces. **Computers Electrical Engineering**, v. 46, p. 328–337, 2015.

CHOLLET, François. **Deep Learning with Python**. Shelter Island, NY, USA: Manning Publications Co., 2017.

COSTA, Rafael D.; MENDES, Juliana P. Artificial Intelligence and Traceability in the Citrus Production Chain. **Brazilian Journal of Technology in Agribusiness**, Embrapa, Brasília, Brazil, v. 18, n. 2, p. 55–70, 2023.

DA COSTA, Arthur Z.; FIGUEROA, Hugo E.H.; FRACAROLLI, Juliana A. Computer vision based detection of external defects on tomatoes using deep learning. **Biosystems Engineering**, v. 190, n. 1, p. 131–144, 2020.

DENG, Jia et al. Imagenet: A large-scale hierarchical image database. **Proceedings of the IEEE conference on computer vision and pattern recognition**, Miami, FL, USA, p. 248–255, 2009.

FERREIRA, Lucas M.; COSTA, Beatriz R. The Impact of Fruit Classification on the Valuation of Brazilian Production. **Journal of Agroindustrial Technology**, Institute of Agroindustrial Technology, Campinas, Brazil, v. 10, n. 1, p. 15–29, 2021.

FORSYTH, David A; PONCE, Jean. A modern approach. **Computer vision: a modern approach**, Prentice-Hall, v. 17, p. 21–48, 2003.

GEDRAITE, E. S.; HADAD, M. Investigation on the effect of a Gaussian Blur in image filtering and segmentation. In: PROCEEDINGS ELMAR-2011. Zadar, Croatia: IEEE, 2011. P. 393–396.

GIACOMETTI, Ellen M. et al. Towards an Automatic Classification of Persian Lime. In: XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2022). Brazil: Sociedade Brasileira de Telecomunicações (SBrT), 2022.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016.

HAN, Hui; WANG, Wen-Yuan; MAO, Bing-Huan. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In_____. **Advances in Intelligent Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. P. 878–887.

HAN, Zewei; WU, Q. M. Jonathan; YANG, Yi. A Survey on Graph Neural Networks and Graph Transformers in Computer Vision: A Task-Oriented Perspective. **arXiv preprint arXiv:2209.13232**, 2022.

HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, I. Textural Features for Image Classification. **IEEE Transactions on Systems, Man, and Cybernetics**, n. 6, p. 610–621, Nov. 1973.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The Elements of Statistical Learning**. New York, NY, USA: Springer, 2009.

HE, Haibo; GARCIA, Edwardo A. Learning from imbalanced data. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 21, n. 9, p. 1263–1284, 2009.

HE, Kaiming et al. Deep residual learning for image recognition. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. Las Vegas, NV, USA: IEEE, 2016. P. 770–778.

HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. **The Definitive Guide to Django: Web Development Done Right**. 2. ed. New York, NY, USA: Apress, 2009.

INGLE, Prashant Devidas; KAUR, Parminder. Adaptive thresholding to robust image binarization for degraded document images. In: 2017 1st International Conference on Intelligent Systems and Information Management (ICISIM). Mumbai, India: IEEE, 2017. P. 189–193.

JAMES, Gareth et al. **An Introduction to Statistical Learning**. New York, NY, USA: Springer-Verlag New York, 2013.

JOANES, D. N.; GILL, C. A. Comparing Measures of Sample Skewness and Kurtosis. **Journal of the Royal Statistical Society: Series D (The Statistician)**, Wiley, v. 47, n. 1, p. 183–189, 1998.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet classification with deep convolutional neural networks. **Communications of the ACM**, v. 60, n. 6, p. 84–90, 2017.

KUMAR, Arvind; SODHI, Sartaj Singh. Comparative Analysis of Gaussian Filter, Median Filter and Denoise Autoenocoder. **International Conference on Computing for Sustainable Global Development**, 2020.

LEMAÎTRE, Guillaume; NOGUEIRA, Fernando; ARIDAS, Christos K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. **Journal of Machine Learning Research**, v. 18, n. 17, p. 1–5, 2017.

LIN, Guichao et al. In-field citrus detection and localisation based on RGB-D image analysis. **Biosystems Engineering**, v. 186, n. 1, p. 34–44, 2019.

LU, Jun et al. Detecting citrus fruits with shadow within tree canopy by a fusing method. In: 2012 5th International Congress on Image and Signal Processing. Chongqing, China: IEEE, 2012. P. 1229–1232.

M. STONEBRAKER, L. A. Rowe; HIROHAMA, M. The implementation of POSTGRES. **Transactions on Knowledge and Data Engineering 2(1), IEEE,** 1990.

NAVIDI, W.C. **Principles of Statistics for Engineers and Scientists**. New York, NY, USA: McGraw-Hill, 2010.

OLIVEIRA, Marina C.; SOUZA, Fernando L. Quality and Commercialization of Tahiti Limes in the International Market. **Journal of Agricultural Science**, Federal University of Viçosa, Viçosa, Brazil, v. 42, n. 4, p. 78–92, 2019.

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

POTHEN, Zania S; NUSKE, Stephen. Texture-based fruit detection via images using the smooth patterns on the fruit. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). Stockholm, Sweden: IEEE, 2016. P. 5171–5176.

SANTOS, Gabriela P.; LIMA, André V. Computer Vision Applied to Agriculture: A Study on Fruit Classification. **Computational Agriculture Journal**, Springer, Berlin, Germany, v. 15, n. 3, p. 101–120, 2022.

SCHMIDHUBER, Jürgen. Deep Learning in Neural Networks: An Overview. **Neural Networks**, Elsevier, v. 61, p. 85–117, 2015.

SERRA, Jean. Image Analysis and Mathematical Morphology: Volume 1: Theoretical Foundations. **Academic Press**, Elsevier, 1982.

SILVA, João P.; ALMEIDA, Ricardo T. Analysis of Tahiti Lime Production in Brazil: Challenges and Perspectives. **Brazilian Journal of Citriculture**, Brazilian Citrus Association, São Paulo, Brazil, v. 35, n. 2, p. 45–60, 2020.

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. Berlin, Germany: Springer, 2010.

TABACHNICK, Barbara G.; FIDELL, Linda S. **Using Multivariate Statistics**. 7th. Hoboken, NJ: Pearson, 2021.

WERBOS, Paul John. **Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences**. Cambridge, MA, USA: Harvard University Press, 1975.

ZAWBAA, Hossam M. et al. Automatic fruit classification using random forest algorithm. In: 2014 14th International Conference on Hybrid Intelligent Systems. Kuwait, Kuwait: IEEE, Dec. 2014. P. 164–168.