

A decorative graphic on the left side of the slide. It features a large light green circle at the top left, a smaller medium green circle at the top center, and three leaf-shaped images. One leaf is large and positioned in the center, showing a close-up of its veins. Another leaf is to its left, showing a white diagonal tear. A third leaf is at the bottom center. A small bright green circle is at the bottom left.

# MODELO DE IDENTIFICAÇÃO DE DOENÇAS AGRÍCOLAS

# RESUMO


- Problemática
- Proposta de solução
- Funcionamento
  - Etapas do Crisp-DM aplicados no projeto
- Próximos passos





# 1. Problemática

Quais as dificuldades do tema abordado?



**Público alvo:** Responsáveis e trabalhadores do setor agrícola ou Pessoas que possuem a própria plantação como fonte de renda.

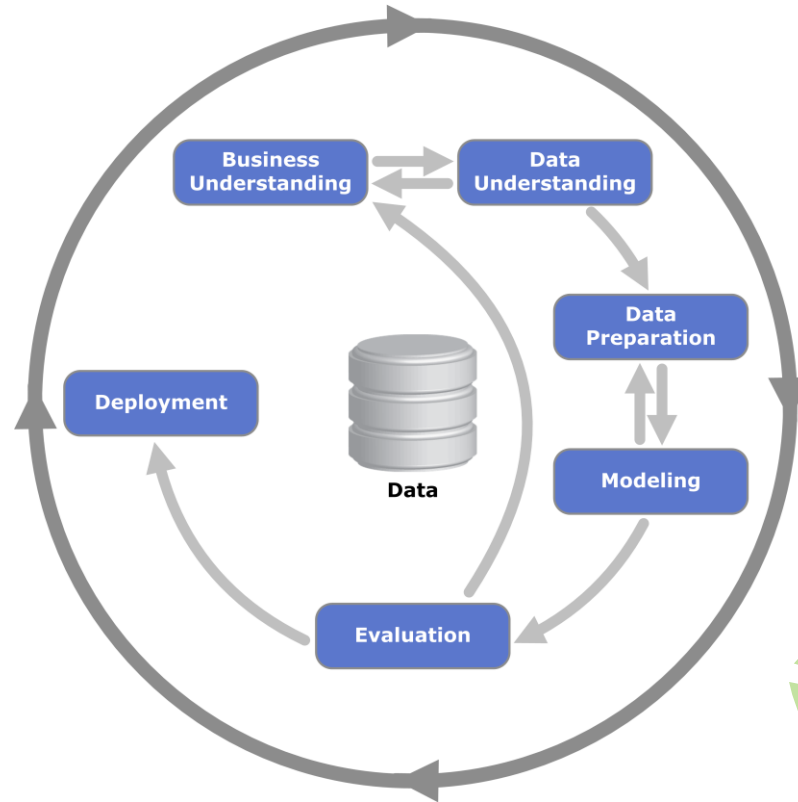
- Dificuldade de controlar manualmente se a plantação está saudável.
- Dificuldade em identificar patologias.
- Alguns produtores não conhecem as possíveis patologias que suas plantações podem sofrer.
- Não existem tecnologias que auxiliem o agricultor nestes momentos. Muitos contam com a experiência mas algumas vezes sem efetividade.
- Com tudo isso, o agricultor pode ter um prejuízo grande o suficiente para ter que encerrar as suas atividades.

The background is a solid light green color. It features several stylized green leaves of different sizes and shades of green. There are also several light green circles of varying diameters scattered across the background. The leaves have visible veins, and the circles are semi-transparent.

## 2. Proposta de solução

Usar um modelo de reconhecimento de imagens para fazer um rápido diagnóstico de doenças agrícolas, para que seja tomada uma ação, visando a recuperação da saúde dessas plantas, além de mitigar a contaminação.

# 3. Funcionamento





### 3.1 – Entendimento dos dados

- Atributos Alvo (case Inicial)
  - Categoria escolhida: Soja
  - Patologia escolhida: Oídio
- Bases de imagens retiradas do site da EMBRAPA.
- Aproximadamente 1.500 amostras, entre saudáveis e doentes.
- Identificar plantas saudáveis ou doentes

## Base de Imagens de Sintomas de Doenças de Plantas (PDDB)

- Termo de uso
- Download de arquivos de imagem compactados
- Publicação relacionada
- Como citar

## Comunidades do repositório

Clique em uma comunidade para ver suas coleções

### Base de Imagens de Sintomas de Doenças de Plantas PDDB

Image Database of Plant Disease Symptoms  
PDDB

### Infopat

Projeto de Integração de processamento digital de imagens em fotografias e sistema especialista para diagnóstico de doenças em plantas no Brasil.

## Busca facetada

### Cultura

Soja (Soybean)	29
Citros (Citrus)	27
Coqueiro (Coconut Tree)	24
Feljao (Dry bean)	24
Mandioca (Cassava)	20
Maracujá (Passion Fruit)	19

### Desordem

Antracnose (Anthracnose)	17
Ferrugem (Rust)	12
Oídio (Powdery mildew)	12
Saudável (Healthy)	7
Mancha Bacteriana (Bacterial spot)	5



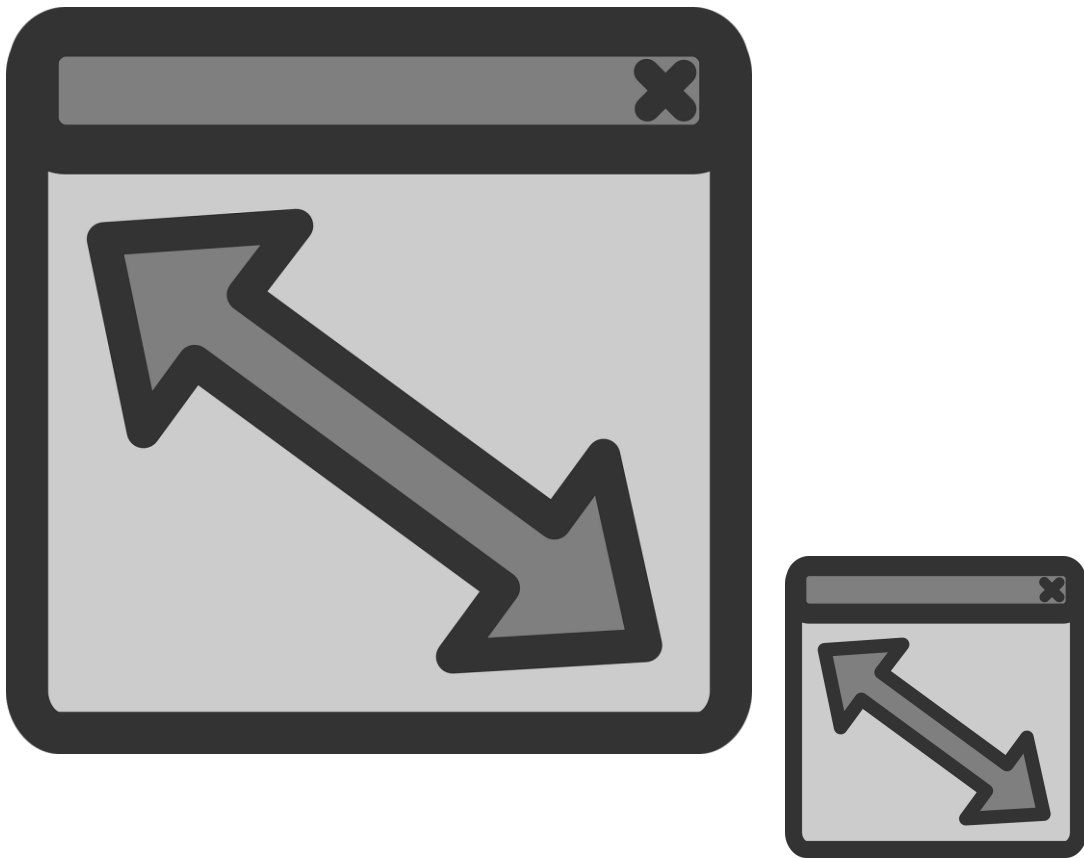


## 3.2 – Pré-processamento

- Identificar algumas características essenciais:
  - posição da folha,
  - evidência da patologia,
  - luminosidade da imagem,
  - densidade das plantas do fundo
- Separação da base entre treinamento, validação e teste
- Redimensionalização
- Dessaturação
- Limiarização/Segmentação

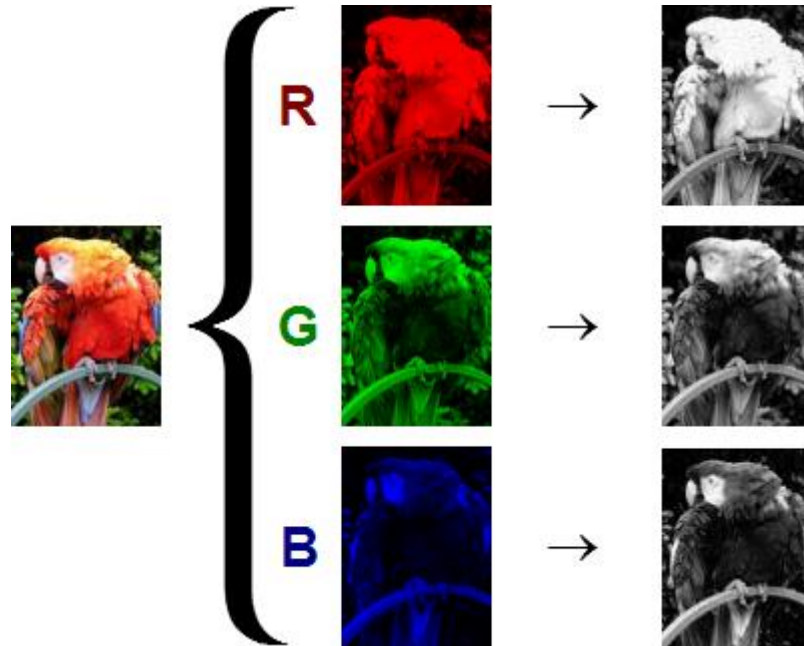
## Redimensionalização:

Diminuir o tamanho de uma imagem para deixar o processamento mais leve e, conseqüentemente, mais rápido, ou, aumentá-la para melhorar a qualidade da imagem e, conseqüentemente, trazer melhores resultados.



## Dessaturação:

É o processo de remoção das camadas RGB para que a limiarização possa ser mais efetiva.



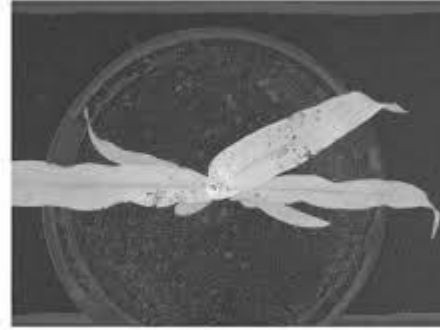
- **Limiarização/Segmentação:** é o processo de segmentação de uma imagem a partir dos tons de cinza de cada objeto.



A.



B.



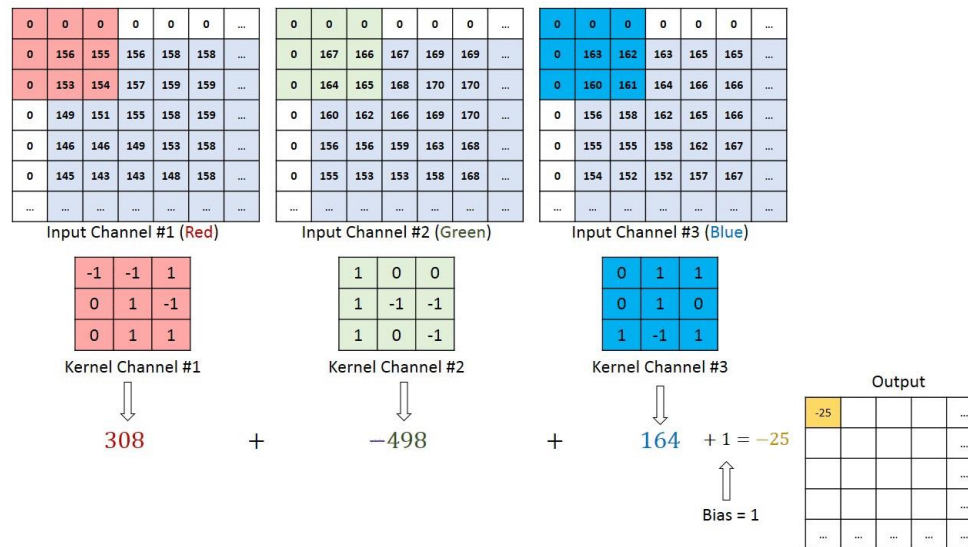


## 3.3 – Modelagem

- Rede Neural Convolucional

Bibliotecas utilizadas: Keras e TensorFlow

- Treinamento do modelo



# Código - Pré-processamento

## Pré-Processamento

In [13]: *# Criando os objetos train\_datagen e validation\_datagen com as regras de pré-processamento das imagens*  
from keras.preprocessing.image import ImageDataGenerator

```
train_datagen = ImageDataGenerator(rescale = 1./255,  
                                   rotation_range=40,  
                                   width_shift_range =0.2,  
                                   height_shift_range =0.2,  
                                   shear_range = 0.2,  
                                   zoom_range = 0.2,  
                                   horizontal_flip = True,  
                                   vertical_flip=True,  
                                   fill_mode='nearest')  
  
validation_datagen = ImageDataGenerator(rescale = 1./255)
```

In [14]: *# Pré-processamento das imagens de treino e validação*  
BATCH\_SIZE = 50

```
training_set = train_datagen.flow_from_directory('C:/Base_Soja_TCC/dataset_treino',  
                                                  target_size = (64, 64),  
                                                  batch_size = 32,  
                                                  class_mode = 'binary')  
  
validation_set = validation_datagen.flow_from_directory('C:/Base_Soja_TCC/dataset_validacao',  
                                                         target_size = (64, 64),  
                                                         batch_size = 32,  
                                                         class_mode = 'binary')
```

Found 1052 images belonging to 2 classes.  
Found 429 images belonging to 2 classes.

## Código - Construção da rede neural Convolutacional

```
In [4]: # Inicializando a Rede Neural Convolutacional
classifier = Sequential()

In [5]: # Passo 1 - Primeira Camada de Convolução
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))

In [6]: # Passo 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Desativa 30% dos neurônios para evitar Overfitting
classifier.add(Dropout(0.3))

In [7]: # Adicionando a Segunda Camada de Convolução
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

In [8]: classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.3))

In [9]: # Passo 3 - Flattening
classifier.add(Flatten())

In [10]: # Passo 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dropout(0.5))

In [11]: # Passo 5 - Camada de saída
classifier.add(Dense(units = 1, activation = 'sigmoid'))

In [12]: # Compilando a rede
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

## 3.4 – Validação

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2)	(None, 31, 31, 32)	0
dropout_1 (Dropout)	(None, 31, 31, 32)	0
conv2d_2 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_2 (MaxPooling2)	(None, 14, 14, 32)	0
dropout_2 (Dropout)	(None, 14, 14, 32)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 128)	802944
dropout_3 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129
Total params: 813,217		
Trainable params: 813,217		
Non-trainable params: 0		



O modelo alcançou 98% de acurácia no treinamento e 99% na validação.  
1052 imagens separadas para treinamento e 429 separadas para validação (classificadas em doente e saudável).



## 3.5 – Demonstração de teste

## Código - Teste (10 imagens de amostragem)

Acurácia: 98%

30 imagens separadas para teste



```
In [75]: # Primeira Imagem
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing import image

test_set = glob.glob('C:/Base_Soja_TCC/dataset_teste/*.jpg')

test_image = np.array([image.img_to_array(image.load_img(image_name, target_size=(64, 64), color_mode='rgb'))/64 for image_name in
training_set.class_indices
y_true = [0,0,0,0,0,1,1,1,1,1]

figure = plt.figure(figsize=(20,8))

for i in range(10):
    if result[i] == 1:
        prediction = 'Saudavel'
    else:
        prediction = 'Doente'

    ax = figure.add_subplot(3, 5, i + 1, ticks=[], yticks=[])
    im = plt.imread(test_set[i])
    ax.imshow(im)
    ax.set_title("{}".format(prediction), color=( "red" if result[i] == y_true[i] else "green"))
```





## 4. Próximos passos

- Para o Trabalho de Conclusão de Curso (TCC) pretendemos treinar o modelo para identificar mais patologias.
- Um outro ponto de melhoria seria, após identificar se existe alguma patologia nas plantas, identificar também qual o tipo da patologia identificada, para assim, auxiliar rapidamente o agricultor com a melhor forma de tratar as plantas contaminadas.



Obrigado!

