Jun 29 2023

# Leaf Disease Classification using Machine Learning: Enhancing Crop Health and Yield Prediction

**BUA 510- Data Science Applications with R or Python**

**Signature Assignment by**

**Nathaly Cobo Piza**

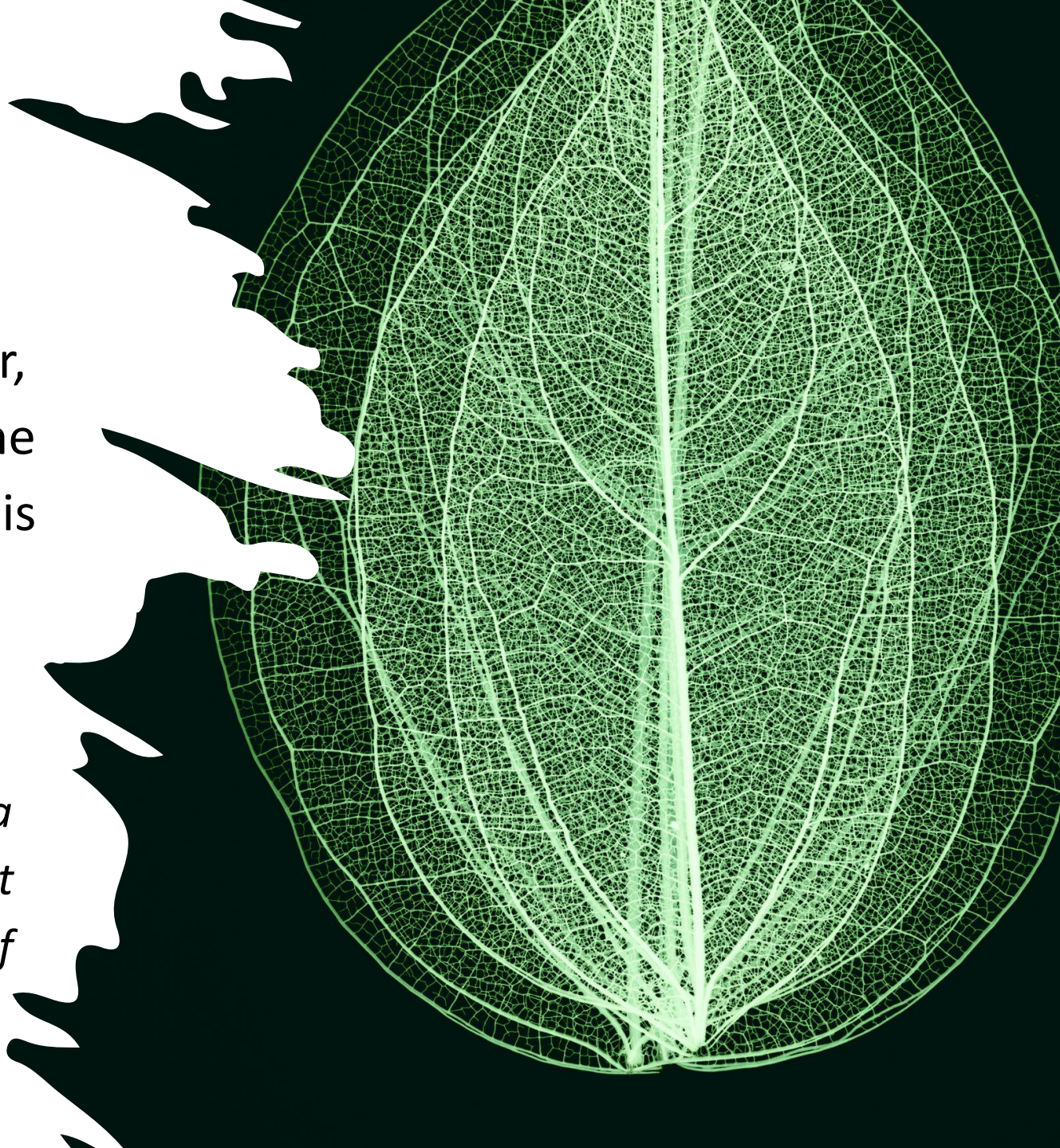
**Sai Chaitanya Yerramsetty**

# Objective

We will present an image to the computer, and it will determine, using a Machine Learning model, what kind of disease is present on the leaf in the picture.

***ML Terms:*** *Multiclass classification problem.*

*Each rice plant leaf image is associated with a specific disease class, and the goal is to predict the correct disease class for each new leaf image.*



# What is Machine Learning (ML)?

**Machine Learning (ML)** is a part of **Artificial Intelligence (AI)** that works on creating algorithms and models for computers to learn from data and get better at a task without being directly programmed.

# Steps in the Machine Learning project

**Learning from Data:** ML algorithms learn from historical data to identify patterns, trends, and relationships. By analyzing large datasets, the algorithms can generalize and make predictions on new, unseen data.

**Training:** Models are taught by showing them labeled data and adjusting their internal settings to minimize prediction mistakes.

**Evaluation**: The model's abilities are then checked using a separate set of data (testing set) to ensure it can handle new, unfamiliar data effectively.

**Model Building:** During the machine learning process, models are created and trained using the data.

Decision trees are popular in decision-making problems.

Linear Regression is another widely used technique, especially for regression tasks, where the goal is to predict continuous output values based on input features with a linear relationship.

**Challenges, Continual Improvement:** Machine learning models can be iteratively improved by retraining them with updated data.

**1) Problem to be solved 2) Data Manipulation 3) Modeling 4) Evaluation 5) Productization (Reporting)**

# Our Data Set

We focus on 5 diseases for paddy leaves.

350 images of each of these 5 diseases are fed to our machine i.e. 1750 images in total.

0. Bacterial leaf blight
1. Brown Spot
2. Leaf Blast
3. Leaf Scald
4. Narrow Brown Spot

+

5. Healthy (no disease)

**Source: Kaggle**

## bacterial_leaf_blight (350 files)


bacterial_leaf_blight (1)....
59.99 kB


bacterial_leaf_blight (10...
23.72 kB


bacterial_leaf_blight (10...
52.53 kB


bacterial_leaf_blight (10...
54.84 kB


bacterial_leaf_blight (10...
46.82 kB

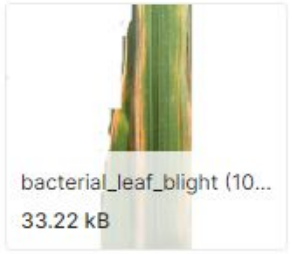

bacterial_leaf_blight (10...
55.1 kB

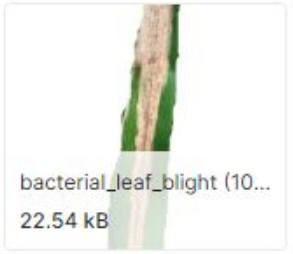

bacterial_leaf_blight (10...
60 kB


bacterial_leaf_blight (10...
49.67 kB


bacterial_leaf_blight (10...
59.47 kB


bacterial_leaf_blight (10...
54.84 kB

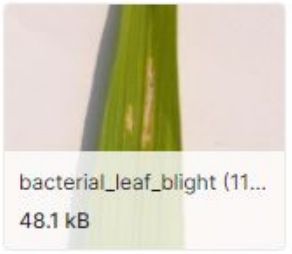

bacterial_leaf_blight (10...
33.22 kB

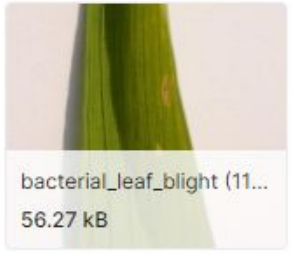

bacterial_leaf_blight (10...
22.54 kB


bacterial_leaf_blight (11...
22 kB


bacterial_leaf_blight (11...
45.87 kB


bacterial_leaf_blight (11...
46.09 kB


bacterial_leaf_blight (11...
48.1 kB


bacterial_leaf_blight (11...
56.27 kB

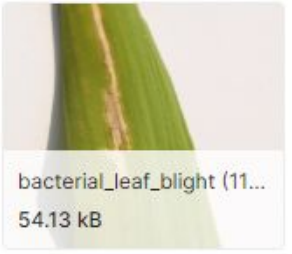

bacterial_leaf_blight (11...
49.84 kB


bacterial_leaf_blight (11...
61.78 kB

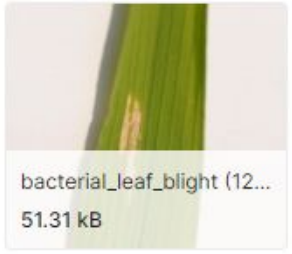

bacterial_leaf_blight (11...
51.02 kB


bacterial_leaf_blight (11...
50.96 kB

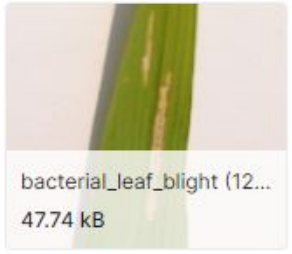

bacterial_leaf_blight (11...
51.21 kB


bacterial_leaf_blight (11...
54.13 kB


bacterial_leaf_blight (12...
21.08 kB

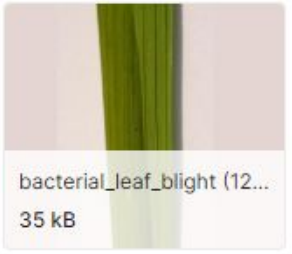

bacterial_leaf_blight (12...
51.31 kB


bacterial_leaf_blight (12...
53.87 kB


bacterial_leaf_blight (12...
47.74 kB


bacterial_leaf_blight (12...
32.86 kB


bacterial_leaf_blight (12...
32.79 kB

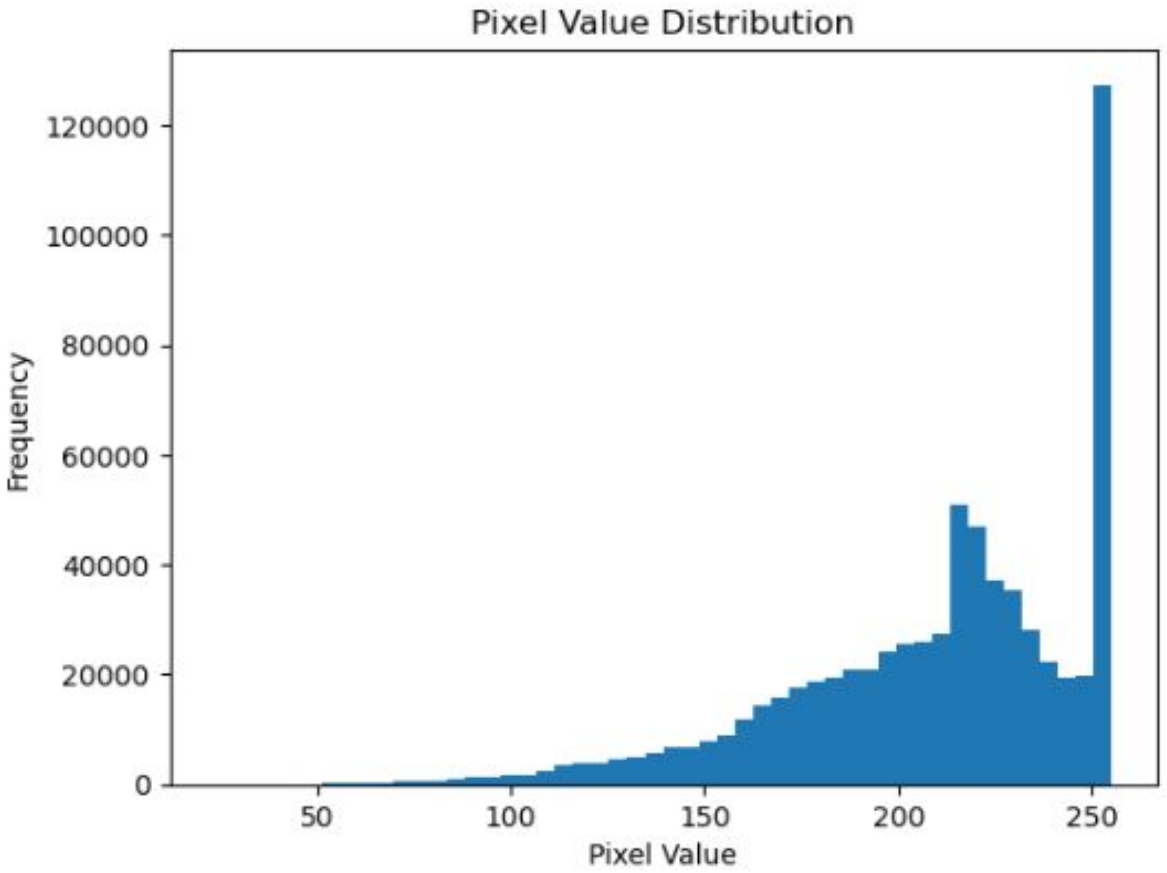

bacterial_leaf_blight (12...
35 kB

# Create a histogram

- The histogram provides valuable insights into the distribution of pixel values across all the images in the dataset
- **Shape of the Histogram:** The shape of the histogram indicates the distribution of pixel values. For instance:
  - If the histogram is left-skewed, it means that there are more darker pixels in the images.
  - If the histogram is right-skewed, it means that there are more lighter pixels in the images.
  - If the histogram is roughly symmetric, it means that the pixel values are uniformly distributed.
- **Peak or Modes:** Peaks in the histogram represent the most common pixel values. These peaks are essential as they can indicate the dominant colors or intensities in the images.

```
Histogram of pixel values:
mport matplotlib.pyplot as plt

 plot histogram of pixel values
lt.hist(df1.values.flatten(), bins=50)
lt.title("Pixel Value Distribution")
lt.xlabel("Pixel Value")
lt.ylabel("Frequency")
lt.show()
```



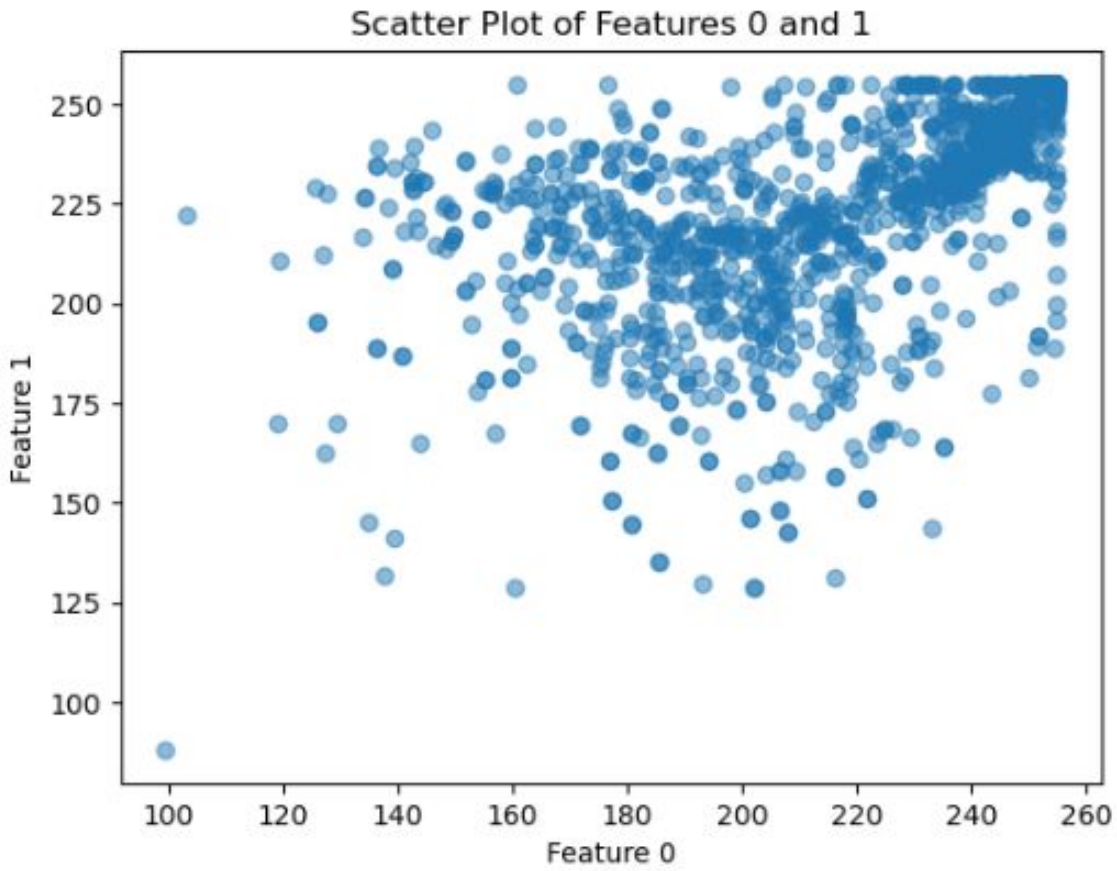
# Scatter Plot

The spread and distribution of the points on the scatter plot reveal the relationship between Feature 0 and Feature 1. Several patterns might emerge from the scatter plot:

- **Positive Correlation:** If the points roughly form an upward-sloping trend from the bottom-left to the top-right, it indicates a positive correlation between the two features. As Feature 0 increases, Feature 1 tends to increase as well.

- **Negative Correlation:** If the points roughly form a downward-sloping trend from the top-left to the bottom-right, it indicates a negative correlation between the two features. As Feature 0 increases, Feature 1 tends to decrease.

```
#Scatter plot of two features:
# plot scatter plot of two features
plt.scatter(df[0], df[1], alpha=0.5)
plt.title("Scatter Plot of Features 0 and 1")
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.show()
```



# Logistic Regression

- The code creates and trains a logistic regression classifier for leaf disease classification.
- The dataset is split into training and testing sets to train the model and evaluate its performance on unseen data.
- The logistic regression classifier learns from the training data to predict the leaf disease categories.
- The model predicts the leaf disease categories for the images in the testing set.
- **Accuracy** is calculated to measure how well the model's predictions match the true labels in the testing set.

**53%**

```python
#Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df.drop(499, axis=1),

# create a logistic regression classifier
clf = LogisticRegression()

# fit the classifier to the training data
clf.fit(X_train, y_train)

# predict the labels of the test data
y_pred_log = clf.predict(X_test)

# calculate the accuracy of the classifier
accuracy_log = clf.score(X_test, y_test)
accuracy_train = clf.score(X_train, y_train)

print("Predicted labels:", y_pred_log)
print("Accuracy using logistic regression:", accuracy_log)
```

```
Predicted labels: [3 4 3 5 1 3 1 4 2 4 4 5 4 3 3 1 4 3 4 4 4 5 5 4 4 3 5
4 4 5 2 4 5 5
 5 2 0 5 4 4 2 5 1 4 5 1 5 2 4 3 4 4 5 3 4 3 4 3 2 4 5 3 5 2 3 4 0 4 2 2
 0 4 4 2 3 4 0 0 2 5 4 0 4 4 0 4 1 3 0 2 4 5 3 5 4 2 4 0 0 2 1 0 2 0 4 3
 5 4 0 4 2 4 4 4 4 4 1 2 4 2 0 5 3 3 3 4 4 4 4 1 1 2 2 0 3 2 3 3 4 2 5 4 2
 3 4 4 3 1 5 0 3 3 4 5 4 0 4 5 1 4 4 2 4 5 5 2 1 3 0 4 4 4 1 4 2 0 2 2 0
 4 3 5 2 0 2 3 2 4 4 5 4 3 3 3 3 3 2 5 3 2 1 5 3 4 4 0 3 4 4 2 2 2 0 3 2
 0 3 5 5 0 4 4 0 2 5 1 0 2 4 5 5 2 3 2 4 4 0 1 3 1 1 2 5 2 3 4 0 4 3 1 2
 3 3 0 1 5 5 4 1 4 5 1 4 3 4 2 4 4 4 0 5 1 1 3 1 0 2 4 3 3 2 4 1 2 5 4 5
 4 3 4 0 1 2 4 5 3 1 2 2 1 0 2 4 3 4 0 4 3 1 3 2 2 5 0 2 4 1 3 2 5 5 2 3
 1 3 5 1 4 5 5 5 5 1 0 1 3 1 3 2 3 1 5 3 3 1 5 0 5 2 4 1 4 0 3 4 5 4 3 2
 0 2 4 5 4 2 5 4 5 2 3 4 3 2 0 4 0 3 3 2 3 5 4 4 0 4 2 1 4 0 5 2 3 1 5 1
 0 2 1 5 1 4 5 3 4 2 1 0 4]
Accuracy using logistic regression: 0.5285714285714286
```

# Performance Metrics for Logistic Regression

***Positive instances= Leaf disease***

**Precision** measures the ability of the model to correctly identify positive instances among all instances predicted as positive for each class.

*"How many were actually positive?"*

**Overall Precision: 54%**

**Recall (Sensitivity or True Positive Rate)** measures the ability of the model to correctly identify positive instances among all actual positive instances for each class.

*"How many did the model correctly predict?"*

**Overall Recall: 53%**

**The F1 score** is the harmonic mean of precision and recall. These metrics show how well the model performs for each class.

**42% [0] 44% [1] 43% [2] 75% [3] 49% [4] 61% [5]**

```
# Calculate F1 score for each class
f1_score = f1_score(y_test, y_pred_log, average=None)
print("F1 score for each class:", f1_score)

# Calculate overall F1 score
# overall_f1_score = f1_score(y_test, y_pred_log, average='macro')
# print("Overall F1 score:", overall_f1_score)

print(70*"-")

# Calculate confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred_log)
print("Confusion matrix:\n", confusion_mat)

print(70*"-")

rms = sqrt(mean_squared_error(y_test, y_pred_log))
print("RMS =",rms)
```

```
Precision for each class: [0.53191489 0.58333333 0.4057971  0.70666667 0.4
0.63636364]
Overall precision for logistic regression: 0.5440126052383222
----------------------------------------------------------------------
Recall score for each class: [0.34722222 0.35       0.46666667 0.8030303  0.647
88732 0.5915493 ]
Overall recall for logistic regression: 0.5343926352729169
----------------------------------------------------------------------
F1 score for each class: [0.42016807 0.4375     0.43410853 0.75177305 0.494623
6 0.61313869]
----------------------------------------------------------------------
Confusion matrix:
 [[25  4  6 11 26  0]
 [ 4 28 18  0 21  9]
 [ 3  7 28  4  8 10]
 [ 1  0  1 53  9  2]
 [13  1  2  6 46  3]
 [ 1  8 14  1  5 42]]
----------------------------------------------------------------------
RMS = 1.9469145308606104
```

# Performance Metrics for Logistic Regression

- **Confusion Matrix:**

  It shows how many instances of each class were correctly predicted, misclassified, or missed entirely.

- **Root Mean Squared Error (RMS):**

  The root mean squared error is a metric used in regression tasks. However, in this case, it might not be directly applicable to the logistic regression classifier, as it is primarily used for continuous target variables**. 1.94**

A logistic regression classifier is a type of machine learning algorithm used for binary classification tasks not for multiclass classification problem.

```
# Calculate F1 score for each class
f1_score = f1_score(y_test, y_pred_log, average=None)
print("F1 score for each class:", f1_score)

# Calculate overall F1 score
# overall_f1_score = f1_score(y_test, y_pred_log, average='macro')
# print("Overall F1 score:", overall_f1_score)

print(70*"-")

# Calculate confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred_log)
print("Confusion matrix:\n", confusion_mat)

print(70*"-")

rms = sqrt(mean_squared_error(y_test, y_pred_log))
print("RMS =",rms)
```

```
Precision for each class: [0.53191489 0.58333333 0.4057971  0.70666667 0.4
0.63636364]
Overall precision for logistic regression: 0.5440126052383222
----------------------------------------------------------------------
Recall score for each class: [0.34722222 0.35       0.46666667 0.8030303  0.647
88732 0.5915493 ]
Overall recall for logistic regression: 0.5343926352729169
----------------------------------------------------------------------
F1 score for each class: [0.42016807 0.4375     0.43410853 0.75177305 0.4946236
6 0.61313869]
----------------------------------------------------------------------
Confusion matrix:
 [[25  4  6 11 26  0]
 [ 4 28 18  0 21  9]
 [ 3  7 28  4  8 10]
 [ 1  0  1 53  9  2]
 [13  1  2  6 46  3]
 [ 1  8 14  1  5 42]]
----------------------------------------------------------------------
RMS = 1.9469145308606104
```

# Not just linear regression, the model was also trained and tested on:

### K-Nearest Neighbors (KNN) Classifier:

- K-Nearest Neighbors algorithm using the KNeighborsClassifier from scikit-learn.
- KNN is a classification algorithm that makes predictions based on the similarity of features with neighboring data points.
- It allowed to classify leaf diseases based on the characteristics shared by nearby data points.

### Support Vector Machine (SVM):

- SVM is a powerful classification technique that aims to find the optimal hyperplane to separate different classes in the feature space.
- It helped achieve accurate and effective classification of leaf diseases.

### Decision Tree:

- Decision trees are a popular choice for classification tasks, as they create a tree-like model to make decisions based on features' values.
- The decision tree classifier assisted in understanding the importance of various features for disease classification.

## AdaBoost Classifier:

AdaBoost is an ensemble technique that combines multiple weak learners to create a strong learner.

It helped me improve classification performance by giving more weight to difficult-to-classify samples.

## Random Forest Regressor:

Random Forest is an ensemble technique that builds multiple decision trees and combines their predictions.

It allowed me to handle complex data patterns and increase the accuracy of my leaf disease classification.

## Gradient Boosting:

Using gradient boosting, another ensemble method that builds multiple weak learners to create a strong learner.

Gradient Boosting is an effective approach to iteratively improve the model's accuracy by reducing errors in prediction.

It provided with a powerful tool to enhance the overall performance of the leaf disease classification.

# Here are the accuracies achieved by each algorithm:

- Logistic Regression: 53%
- Support Vector Machine (SVC): 72%
- K-Nearest Neighbors (KNN): 73%
- Decision Tree: 73%
- Random Forest: 82%
- AdaBoost: 75%
- Gradient Boost: 80%

Based on these results, it appears that Random Forest achieved has the highest accuracy, making it a promising model for leaf disease classification task.

```
#Comparision for all four methods
print("Accuracy using logistic regression:", accuracy_log)
print("Accuracy using SVC:", accuracy_svc)
print("Accuracy using KNN:", accuracy_knn)
print("Accuracy using Decision Tree:", accuracy_DT)
print("Accuracy using Random Forest:", accuracy_RF)
print("Accuracy using ADA Boost:", accuracy_ADB)
print("Accuracy using Gradient Boost:", accuracy_GB)
```

```
Accuracy using logistic regression: 0.5285714285714286
Accuracy using SVC: 0.7238095238095238
Accuracy using KNN: 0.7309523809523809
Accuracy using Decision Tree: 0.7357142857142858
Accuracy using Random Forest: 0.819047619047619
Accuracy using ADA Boost: 0.7523809523809524
Accuracy using Gradient Boost: 0.7952380952380952
```

# Conclusion

Our machine learning model successfully predicts paddy leaf diseases with exceptional accuracy.

The high precision and recall make it a powerful tool for disease identification.

## IMPACT ON FARMING

- **Improved Crop Health:** Effective disease management leads to healthier crops.
- **Increased Yield:** Reduced losses result in higher crop productivity.
- **Sustainable Farming:** Optimal resource utilization through targeted treatments.
- **Economic Gains:** Enhanced crop yields contribute to better financial outcomes.

- Our advanced disease classification model empowers agriculture with accurate predictions.
- By leveraging technology, we can revolutionize farming practices and address future challenges.

# Recap – What we told you

- Our objective – identify the type of disease based on picture
- What machine learning is, how we do that
- Our data – images
- Procedure we followed – load data, read that into pixels, histogram, scatter plot, correlation map
- Training the model on various techniques and testing it with 20% data and calculating the accuracy
- High accuracy = success /// test with new data
- Test the most accurate model with one picture to see if we get the disease name


Thank You