Documents & Results of the code

Github Link: https://github.com/HEGAZYI/plant-Disease-detection---computer-vision-project

Plant Disease Detection Using Deep Learning

This project aims to classify corn leaf conditions into four categories using deep learning techniques. The conditions include:

1. Corn Northern Leaf Blight
2. Corn Healthy
3. Corn Gray Leaf Spot
4. Corn Common Rust

The project implements a convolutional neural network (CNN) For image classification And applies data augmentation to improve model generalization

Step 1: Import libraries

```
# Core Libraries
import numpy as np
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import cv2
import os
import glob as gb
import pandas as pd

# TensorFlow Utilities
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.layers import RandomFlip, RandomRotation, RandomZoom, RandomHeight, RandomWidth
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D, LeakyReLU

#sklearn usage
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns

#logistic regression classifier
from sklearn.linear_model import LogisticRegression
```

Step 2: Dataset Preparation

Define the Dataset Directory:

The dataset includes images categorized into folders by disease type.

Load the Dataset:

Use TensorFlow's image_dataset_from_directory to load and preprocess the dataset

```
Found 3852 files belonging to 4 classes.
Using 3467 files for training.
--------------------------------------------------
Found 3852 files belonging to 4 classes.
Using 385 files for validation.
--------------------------------------------------
['Corn___Common_Rust', 'Corn___Gray_Leaf_Spot', 'Corn___Healthy', 'Corn___Northern_Leaf_Blight']
```

Step 3: Analyze Data Imbalance

Visualize the distribution of samples across classes to identify potential imbalances.

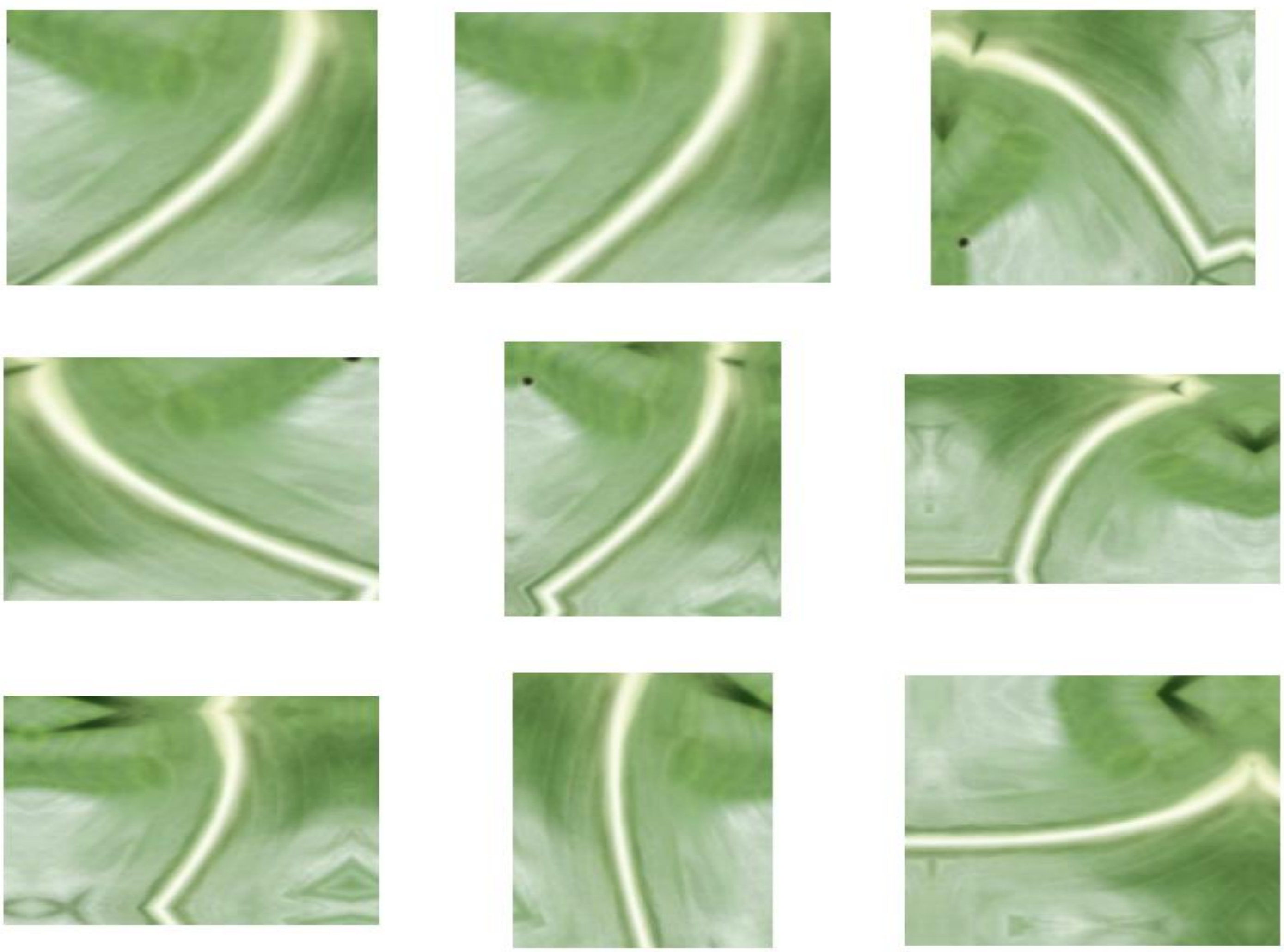
Step 4: Compute Class Weights

Handle data imbalance by calculating class weights.

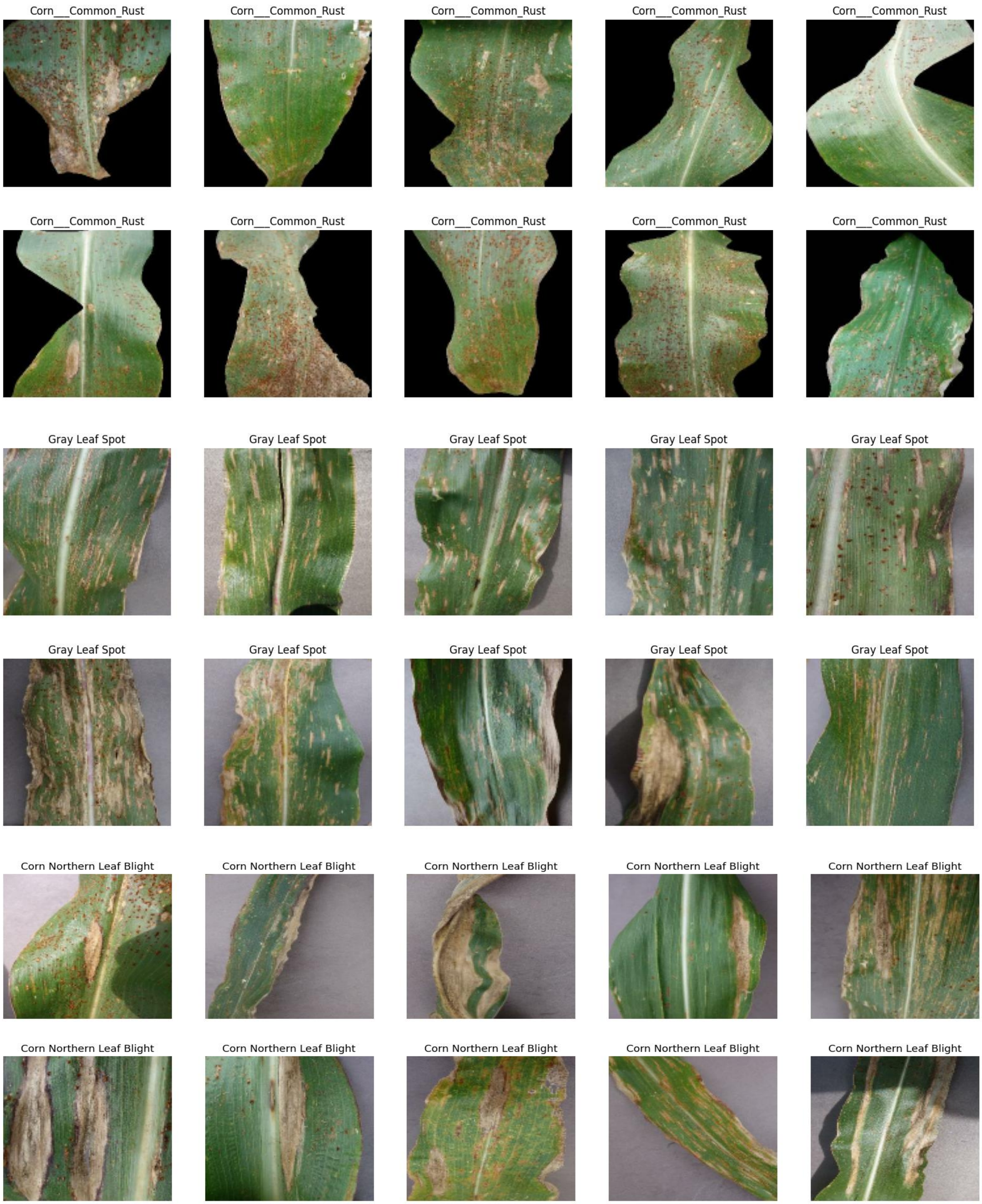
```
Weight for class 0: 0.81
Weight for class 1: 1.88
Weight for class 2: 0.83
Weight for class 3: 0.98
```

Step 5: Data Augmentation

Enhance dataset variability using augmentation techniques like flipping, rotation, and zooming.



Step 6: Display sample images from each class



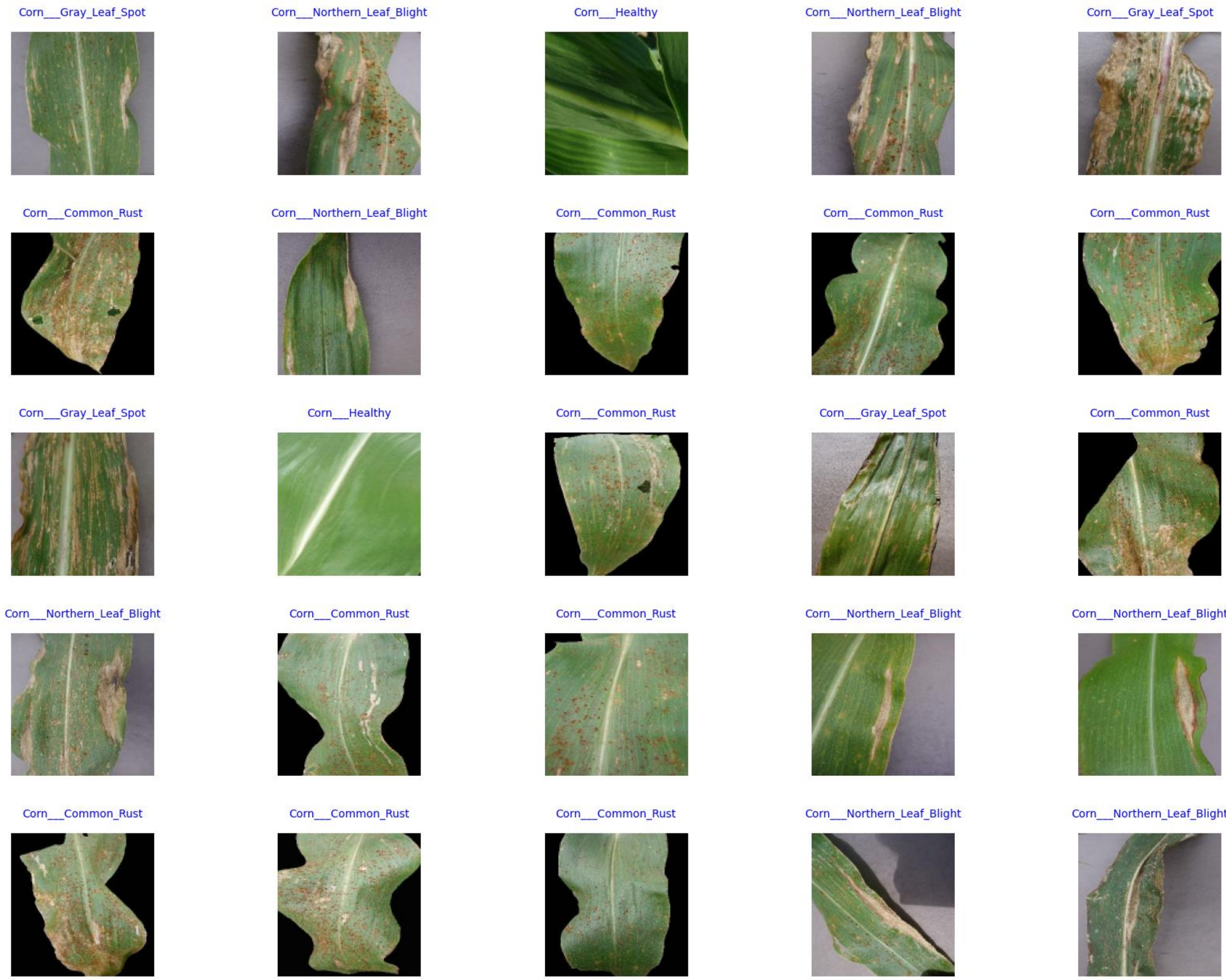
Step 7: define the model

Build a CNN model for classification.

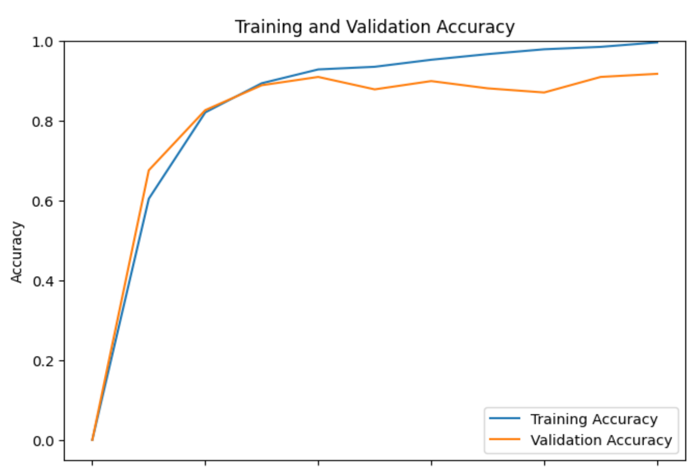
Step 8: train the model

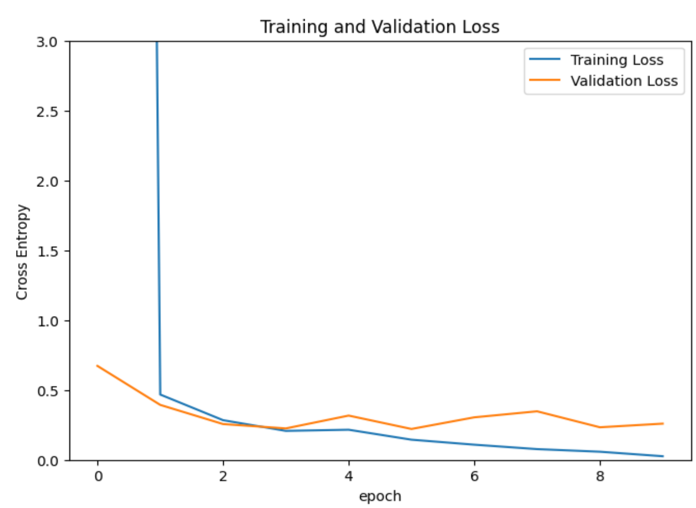
```
Total params: 3,305,156 (12.61 MB)

Trainable params: 3,305,156 (12.61 MB)

Non-trainable params: 0 (0.00 B)

Epoch 1/10
28/28 - 27s - 970ms/step - accuracy: 0.6040 - loss: 47.4613 - val_accuracy: 0.6753 - val_loss: 0.6721
Epoch 2/10
28/28 - 23s - 821ms/step - accuracy: 0.8203 - loss: 0.4673 - val_accuracy: 0.8260 - val_loss: 0.3932
Epoch 3/10
28/28 - 20s - 711ms/step - accuracy: 0.8933 - loss: 0.2838 - val_accuracy: 0.8883 - val_loss: 0.2560
Epoch 4/10
28/28 - 20s - 712ms/step - accuracy: 0.9279 - loss: 0.2072 - val_accuracy: 0.9091 - val_loss: 0.2248
Epoch 5/10
28/28 - 20s - 716ms/step - accuracy: 0.9345 - loss: 0.2154 - val_accuracy: 0.8779 - val_loss: 0.3167
Epoch 6/10
28/28 - 20s - 710ms/step - accuracy: 0.9521 - loss: 0.1441 - val_accuracy: 0.8987 - val_loss: 0.2208
Epoch 7/10
28/28 - 20s - 709ms/step - accuracy: 0.9663 - loss: 0.1083 - val_accuracy: 0.8805 - val_loss: 0.3041
Epoch 8/10
28/28 - 20s - 711ms/step - accuracy: 0.9784 - loss: 0.0767 - val_accuracy: 0.8701 - val_loss: 0.3473
Epoch 9/10
28/28 - 20s - 719ms/step - accuracy: 0.9844 - loss: 0.0580 - val_accuracy: 0.9091 - val_loss: 0.2330
Epoch 10/10
28/28 - 20s - 721ms/step - accuracy: 0.9957 - loss: 0.0257 - val_accuracy: 0.9169 - val_loss: 0.2590
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 147ms/step - accuracy: 0.9227 - loss: 0.2521

[0.2589676082134247, 0.916883111000061]
```

Step 9: Visualize predictions



Step 10: Evaluate Performance



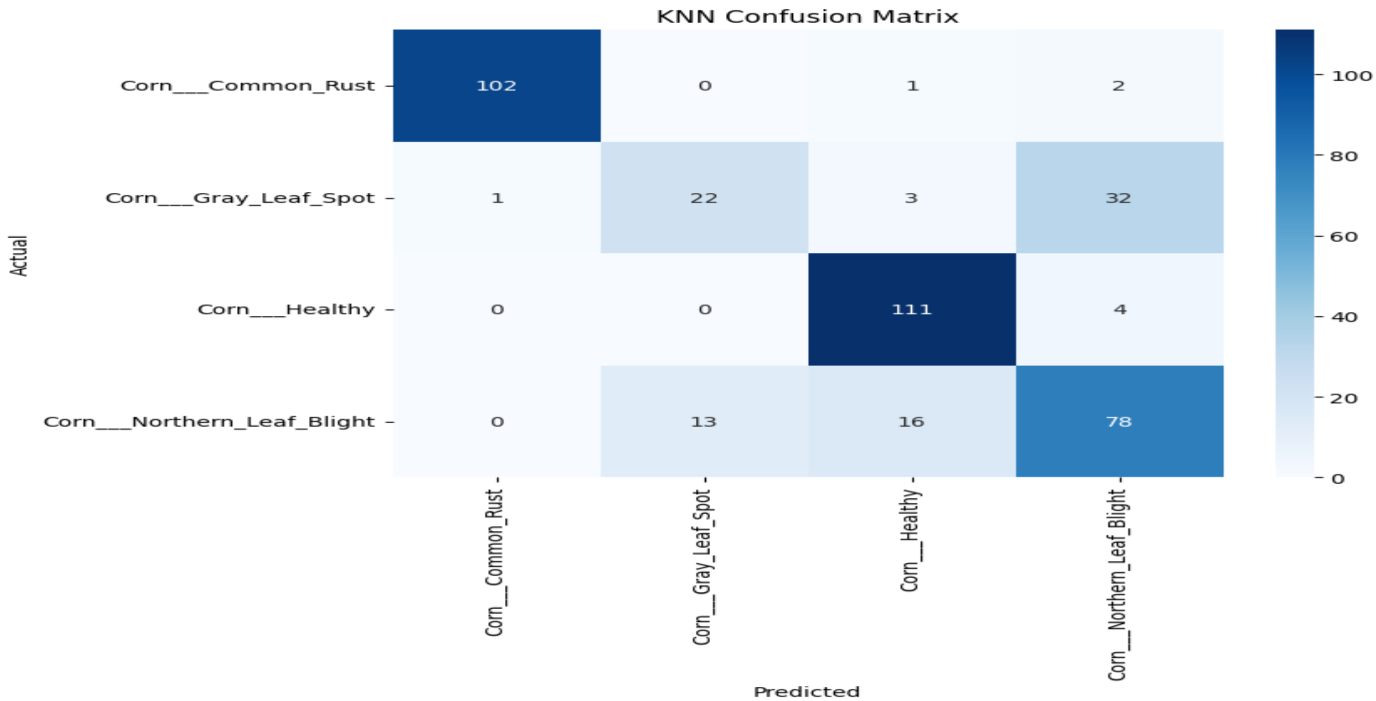


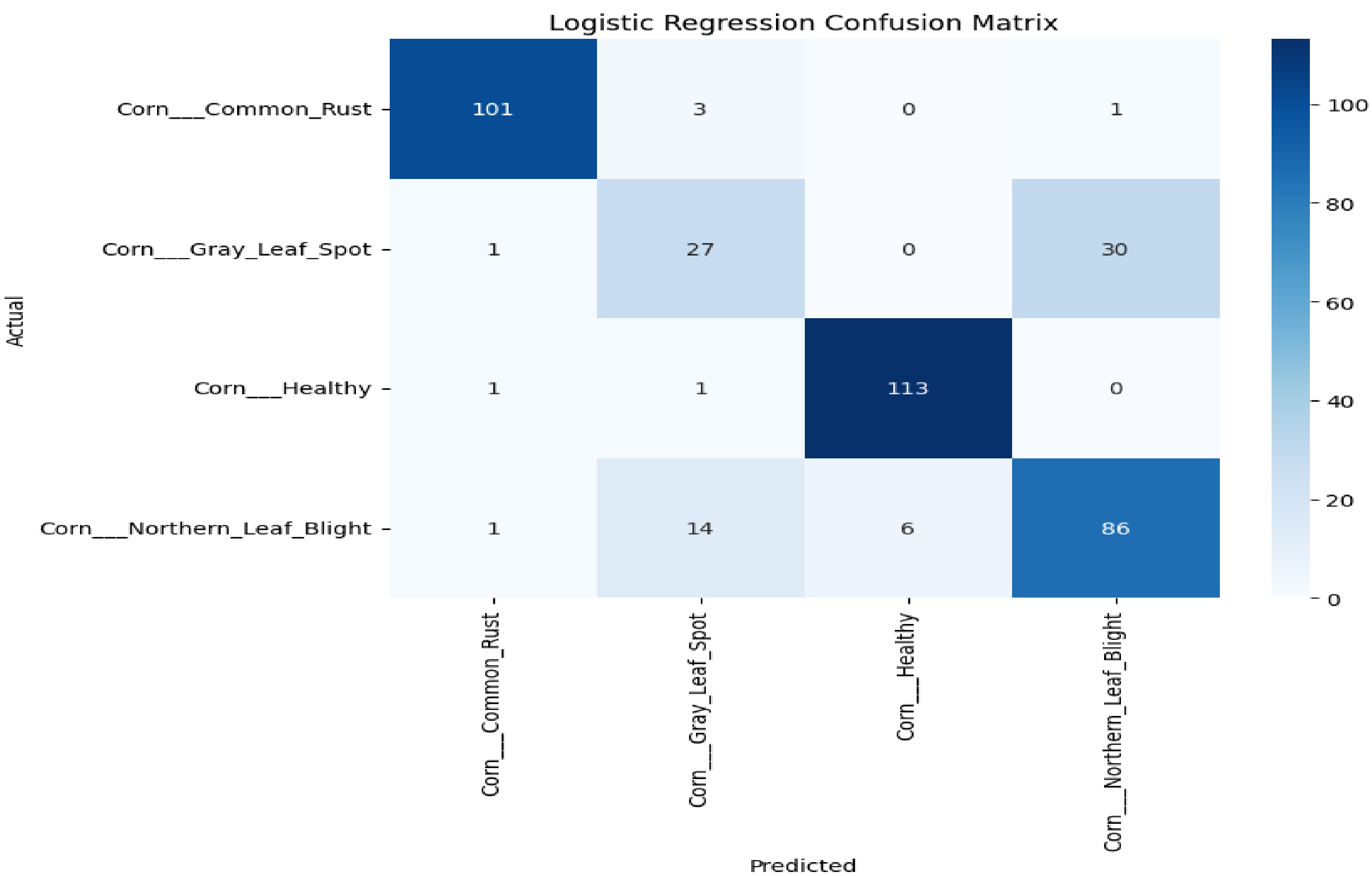
Step 11: Train and Evaluate Machine Learning Models

```
KNN Accuracy: 0.812987012987013
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       105
           1       0.63      0.38      0.47        58
           2       0.85      0.97      0.90       115
           3       0.67      0.73      0.70       107

    accuracy                           0.81       385
   macro avg       0.78      0.76      0.76       385
weighted avg       0.80      0.81      0.80       385
```

```
Logistic Regression Accuracy: 0.8493506493506493
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.97       105
           1       0.60      0.47      0.52        58
           2       0.95      0.98      0.97       115
           3       0.74      0.80      0.77       107

    accuracy                           0.85       385
   macro avg       0.81      0.80      0.81       385
weighted avg       0.84      0.85      0.84       385
```

Step 12: Visualize Confusion Matrices





الحمد لله الذي هدانا لهذا