In [ ]:

In [1]:
```
import cv2
import numpy as np
import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

In [2]:
```
def preprocess_image(image_path, size=(128, 128)):

    img = cv2.imread(image_path)

    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    img = cv2.resize(img, size)

    img = img / 255.0

    return img
```

In [3]:
```
# Define CNN model
def create_cnn_model():
    model = Sequential()

    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
    model.add(MaxPooling2D((2, 2)))  # Max pooling layer

    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))

    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))

    model.add(Flatten())

    model.add(Dense(128, activation='relu'))

    model.add(Dropout(0.5))

    model.add(Dense(38, activation='softmax'))

    model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    return model
```

In [5]:
```
pip install split-folders
```

```
Requirement already satisfied: split-folders in c:\users\hp\anaconda3\lib\site-packages (0.5.1)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
import os
input_dir = r"C:\Users\HP\Desktop\DL WORK\PlantVillage"
print(os.listdir(input_dir))
```

```
['Pepper__bell___Bacterial_spot', 'Pepper__bell___healthy', 'PlantVillage', 'Potato___Early_blight', 'Potato___healthy', 'Potato___Late_blight', 'Tomato_Bacterial_spot', 'Tomato_Early_blight', 'Tomato_healthy', 'Tomato_Late_blight', 'Tomato_Leaf_Mold', 'Tomato_Septoria_leaf_spot', 'Tomato_Spider_mites_Two_spotted_spider_mite', 'Tomato__Target_Spot', 'Tomato__Tomato_mosaic_virus', 'Tomato__Tomato_YellowLeaf__Curl_Virus']
```

In [5]:
```
import splitfolders

input_dir = r"C:\Users\HP\Desktop\DL WORK\PlantVillage"

splitfolders.ratio(input_dir, output="PlantVillage_split", seed=42, ratio=(.7, .2, .1))
```

```
Copying files: 20639 files [01:05, 316.07 files/s]
```

In [9]:
```
import os
print(os.getcwd())
```

```
C:\Users\HP\Desktop\DL PRO NEW
```

In [10]:
```
train_dir = r"C:\Users\HP\Desktop\DL PRO NEW\PlantVillage_split\train"
val_dir = r"C:\Users\HP\Desktop\DL PRO NEW\PlantVillage_split\val"
test_dir = r"C:\Users\HP\Desktop\DL PRO NEW\PlantVillage_split\test"

train_datagen = ImageDataGenerator(rescale=1.0/255.0,
                                   rotation_range=20,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

val_datagen = ImageDataGenerator(rescale=1.0/255.0)

test_datagen = ImageDataGenerator(rescale=1.0/255.0)

train_data = train_datagen.flow_from_directory(train_dir,
                                               target_size=(128, 128),
                                               batch_size=32,
                                               class_mode='sparse')

val_data = val_datagen.flow_from_directory(val_dir,
                                           target_size=(128, 128),
                                           batch_size=32,
                                           class_mode='sparse')

test_data = test_datagen.flow_from_directory(test_dir,
                                             target_size=(128, 128),
                                             batch_size=32,
                                             class_mode='sparse',
                                             shuffle=False)
```

```
Found 16504 images belonging to 16 classes.
Found 6198 images belonging to 16 classes.
Found 2076 images belonging to 16 classes.
```

In [11]:
```
#Train the Model
```

In [12]:
```
model = create_cnn_model()

from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(train_data,
                    epochs=25,
                    validation_data=val_data,
                    callbacks=[early_stop])

model.save("plant_leaf_disease_model.h5")
```

```
C:\Users\HP\anaconda3\lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\HP\anaconda3\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 297s 567ms/step - accuracy: 0.2629 - loss: 2.4453 - val_accuracy: 0.5463 - val_loss: 1.4324
Epoch 2/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 411s 798ms/step - accuracy: 0.5593 - loss: 1.3497 - val_accuracy: 0.5794 - val_loss: 1.3633
Epoch 3/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 205s 396ms/step - accuracy: 0.6308 - loss: 1.0962 - val_accuracy: 0.7728 - val_loss: 0.6333
Epoch 4/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 1155s 2s/step - accuracy: 0.6853 - loss: 0.9578 - val_accuracy: 0.7251 - val_loss: 0.7997
Epoch 5/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 353s 684ms/step - accuracy: 0.7227 - loss: 0.8117 - val_accuracy: 0.7988 - val_loss: 0.6279
Epoch 6/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 191s 369ms/step - accuracy: 0.7302 - loss: 0.7993 - val_accuracy: 0.7438 - val_loss: 0.8676
Epoch 7/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 194s 375ms/step - accuracy: 0.7596 - loss: 0.7227 - val_accuracy: 0.7641 - val_loss: 0.7225
Epoch 8/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 201s 389ms/step - accuracy: 0.7897 - loss: 0.6287 - val_accuracy: 0.8235 - val_loss: 0.5179
Epoch 9/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 209s 406ms/step - accuracy: 0.8038 - loss: 0.5903 - val_accuracy: 0.8185 - val_loss: 0.5586
Epoch 10/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 211s 408ms/step - accuracy: 0.8010 - loss: 0.5728 - val_accuracy: 0.8700 - val_loss: 0.4078
Epoch 11/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 216s 419ms/step - accuracy: 0.8198 - loss: 0.5326 - val_accuracy: 0.8890 - val_loss: 0.3157
Epoch 12/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 209s 404ms/step - accuracy: 0.8304 - loss: 0.4920 - val_accuracy: 0.8582 - val_loss: 0.4707
Epoch 13/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 203s 394ms/step - accuracy: 0.8347 - loss: 0.5075 - val_accuracy: 0.9021 - val_loss: 0.2798
Epoch 14/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 203s 394ms/step - accuracy: 0.8434 - loss: 0.4685 - val_accuracy: 0.9311 - val_loss: 0.2152
Epoch 15/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 202s 391ms/step - accuracy: 0.8585 - loss: 0.4343 - val_accuracy: 0.8942 - val_loss: 0.3264
Epoch 16/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 204s 395ms/step - accuracy: 0.8585 - loss: 0.4286 - val_accuracy: 0.8625 - val_loss: 0.4615
Epoch 17/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 202s 391ms/step - accuracy: 0.8632 - loss: 0.4074 - val_accuracy: 0.9067 - val_loss: 0.2726
Epoch 18/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 202s 391ms/step - accuracy: 0.8636 - loss: 0.4222 - val_accuracy: 0.8540 - val_loss: 0.4925
Epoch 19/25
516/516 ━━━━━━━━━━━━━━━━━━━━ 1061s 2s/step - accuracy: 0.8743 - loss: 0.3838 - val_accuracy: 0.8961 - val_loss: 0.3932
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

In [13]:
```
test_loss, test_accuracy = model.evaluate(test_data)
print(f"\nTest Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
65/65 ━━━━━━━━━━━━━━━━━━━━ 9s 135ms/step - accuracy: 0.9434 - loss: 0.1831

Test Loss: 0.2188
Test Accuracy: 0.9316
```

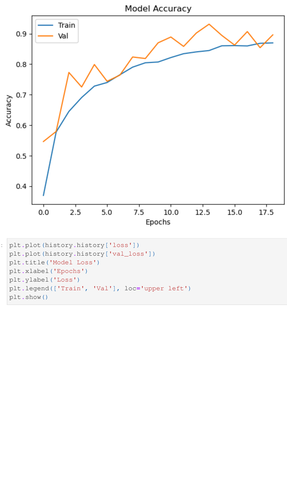
In [12]:
```
#Test the Model on New Images
```

In [14]:
```
# import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
# import os

model = load_model("plant_leaf_disease_model.h5")

class_indices = train_data.class_indices
idx_to_class = {v: k for k, v in class_indices.items()}

def preprocess_image(image_path):
    img = image.load_img(image_path, target_size=(128, 128))
    img_array = image.img_to_array(img) / 255.0  # Normalize pixel values
    return img_array

def predict_leaf_disease(image_path):
    img = preprocess_image(image_path)
    img = np.expand_dims(img, axis=0)
    prediction = model.predict(img)
    class_idx = np.argmax(prediction)
    class_name = idx_to_class[class_idx]
    return class_name

test_image_path = r"C:\Users\HP\Desktop\DL WORK\new.jpg"
predicted_class = predict_leaf_disease(test_image_path)

print(f"The predicted class is: {predicted_class}")
if "healthy" in predicted_class.lower():
    print("The leaf is HEALTHY ✅")
else:
    print(f"The leaf is DISEASED ❌ ")
```

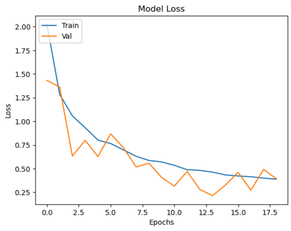
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 290ms/step
The predicted class is: Potato___healthy
The leaf is HEALTHY ✅
```

In [ ]:
```
#visualizing the model's training performance over time.
```

In [19]:
```
# Plot training & validation accuracy and loss
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```



In [18]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```


Model Loss
Train
Val
Loss
Epochs
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5