In [1]:
```
#!pip install matplotlib
import os
import cv2
import numpy as np
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split
```

In [2]:
```
def extract_features(img):

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    resized = cv2.resize(gray, (250, 250))
    features = resized.flatten()
    return features
```

In [3]:
```
path='C:\\Users\\KIIT\\Desktop\\DATASET\\train2'
```

In [4]:
```
classes=['Bacterial_spot',
 'Early_blight',
 'Late_blight',
 'Leaf_Mold',
 'Septoria_leaf_spot',
 'Spider_mites Two-spotted_spider_mite',
 'Target_Spot',
 'Tomato_Yellow_Leaf_Curl_Virus',
 'Tomato_mosaic_virus',
 'healthy',
 'powdery_mildew']
```

In [5]:

```
count=0
data = []
labels = []
for cls in classes:
    folder_path = os.path.join(path, cls)
    for img_name in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img_name)
        # Check if the file is an image file
        if img_path.endswith('.png') or img_path.endswith('.jpg') or img_path.endswith('.jpeg'):
            img = cv2.imread(img_path)
            if img is not None:
                #print("Loaded image:", img_path, "Dimensions:", img.shape, "Type:", img.dtype)
                count=count+1
                features = extract_features(img)
                data.append(features)
                labels.append(cls)
            else:
                print("Error: Failed to load image:", img_path)
        else:
            print("Skipping non-image file:", img_path)
```

```
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\00416648-be6e-4bd4-bc8d
-82f43f8a7240___GCREC_Bact.Sp 3110.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\0045ba29-ed1b-43b4-afde
-719cc7adefdb___GCREC_Bact.Sp 6254.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\00639d29-2d1a-4fcf-9bd3
-a2b3109c74c4___UF.GRC_BS_Lab Leaf 1054.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\00728f4d-83a0-49f1-87f8
-374646fcda05___GCREC_Bact.Sp 6326.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\00a7c269-3476-4d25-b744
-44d6353cd921___GCREC_Bact.Sp 5807.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\00b7e89a-e129-4576-b51f
-48923888bff9___GCREC_Bact.Sp 6202.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\01375198-62af-4c40-bddf
-f3c11107200b___GCREC_Bact.Sp 5914.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\01a3cf3f-94c1-44d5-8972
-8c509d62558e___GCREC_Bact.Sp 3396.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\01a46cb5-d354-4f59-868e
-e56186701541___GCREC_Bact.Sp 5638.JPG
Skipping non-image file: C:\Users\KIIT\Desktop\DATASET\train2\Bacterial_spot\01e079ba-939a-4681-8983
```

In [6]:
```
count
```

Out[6]: 524

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
```

In [8]:
```
modle = DecisionTreeClassifier(random_state=42)
modle.fit(X_train, y_train)
```

Out[8]: DecisionTreeClassifier(random_state=42)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:
```
y_pred = modle.predict(X_test)
```

In [10]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6952380952380952
```

In [11]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [12]:
```

X_train = np.array(X_train)
y_test = np.array(y_test)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train.reshape(X_train.shape[0], -1), y_train)
```

Out[12]: RandomForestClassifier(random_state=42)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]:
```
model.score(X_test,y_test)
```

Out[13]: 0.7619047619047619

In [14]:
```
from sklearn.svm import SVC
```

In [15]:
```
model2=SVC()
```

In [16]:
```
model2.fit(X_train,y_train)
```

Out[16]: SVC()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [17]:
```
model2.score(X_test,y_test)
```

Out[17]: 0.7619047619047619

In [18]:
```
import tensorflow as tf
from tensorflow.keras import models,layers
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
```

In [22]:
```
train_data_gen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
test_data_gen = ImageDataGenerator(rescale=1./255)
train_generator = train_data_gen.flow_from_directory(
    'C:\\Users\\KIIT\\Desktop\\DATASET\\train2',
    target_size=(256, 256),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)
```

```
Found 4638 images belonging to 11 classes.
```

In [23]:
```
validation_generator = train_data_gen.flow_from_directory(
    'C:\\Users\\KIIT\\Desktop\\DATASET\\train2',
    target_size=(256, 256),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 1154 images belonging to 11 classes.
```

In [32]:
```
test_generator = test_data_gen.flow_from_directory(
    'C:\\Users\\KIIT\\Desktop\\DATASET\\train2',
    target_size=(256, 256),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 5792 images belonging to 11 classes.
```

In [25]:

```
CNN = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(256,256,3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(100, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(11, activation='softmax')
])

CNN.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [26]:
```
history = CNN.fit(
    train_generator,

    epochs=25,
    batch_size=32,
    validation_data=validation_generator
)
```

```
Epoch 1/25
145/145 [==============================] - 883s 6s/step - loss: 1.8004 - accuracy: 0.3816 - val_los
s: 1.1871 - val_accuracy: 0.6187
Epoch 2/25
145/145 [==============================] - 624s 4s/step - loss: 1.0193 - accuracy: 0.6404 - val_los
s: 0.8893 - val_accuracy: 0.7080
Epoch 3/25
145/145 [==============================] - 636s 4s/step - loss: 0.8118 - accuracy: 0.7139 - val_los
s: 0.8541 - val_accuracy: 0.7114
Epoch 4/25
145/145 [==============================] - 728s 5s/step - loss: 0.5982 - accuracy: 0.7913 - val_los
s: 0.7581 - val_accuracy: 0.7582
Epoch 5/25
145/145 [==============================] - 686s 5s/step - loss: 0.4239 - accuracy: 0.8549 - val_los
s: 0.6737 - val_accuracy: 0.7929
Epoch 6/25
145/145 [==============================] - 651s 4s/step - loss: 0.2818 - accuracy: 0.9034 - val_los
s: 0.7070 - val_accuracy: 0.7981
Epoch 7/25
```

In [33]:
```
test_loss, test_acc = CNN.evaluate(test_generator)
```

```
181/181 [==============================] - 208s 1s/step - loss: 0.2710 - accuracy: 0.9579
```

In [34]:
```
test_acc
```

Out[34]:
```
0.9578729271888733
```

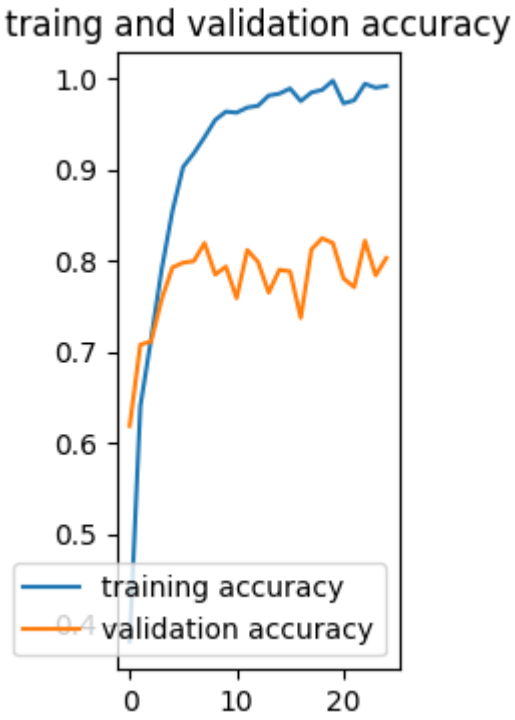
In [35]:
```
print(history.history.keys())
```

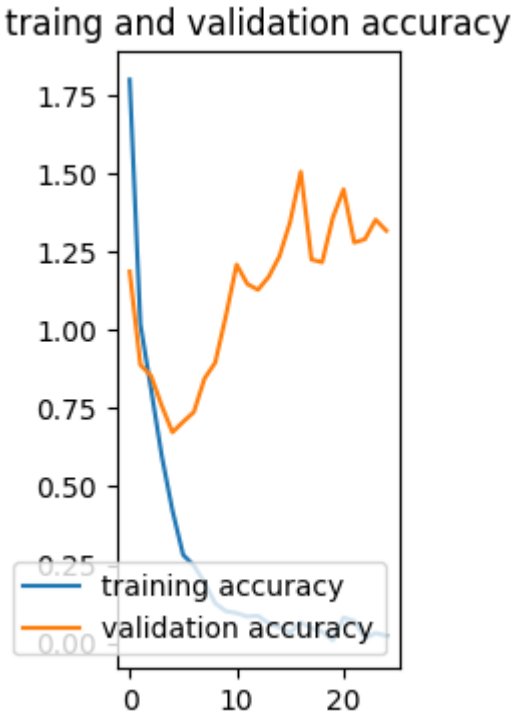
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [36]:
```
acc=history.history['accuracy']
val_acc=history.history['val_accuracy']

loss=history.history['loss']
val_loss=history.history['val_loss']
```

In [44]:
```
epochs=25
plt.figure(figsize=(4,4))
plt.subplot(1,2,1)
plt.plot(range(epochs),acc,label='training accuracy')
plt.plot(range(epochs),val_acc,label='validation accuracy')
plt.legend(loc='lower right')
plt.title('traing and validation accuracy')

plt.figure(figsize=(4,4))
plt.subplot(1,2,1)
plt.plot(range(epochs),loss,label='training accuracy')
plt.plot(range(epochs),val_loss,label='validation accuracy')
plt.legend(loc='lower right')
plt.title('traing and validation accuracy')
```

Out[44]:
```
Text(0.5, 1.0, 'traing and validation accuracy')
```

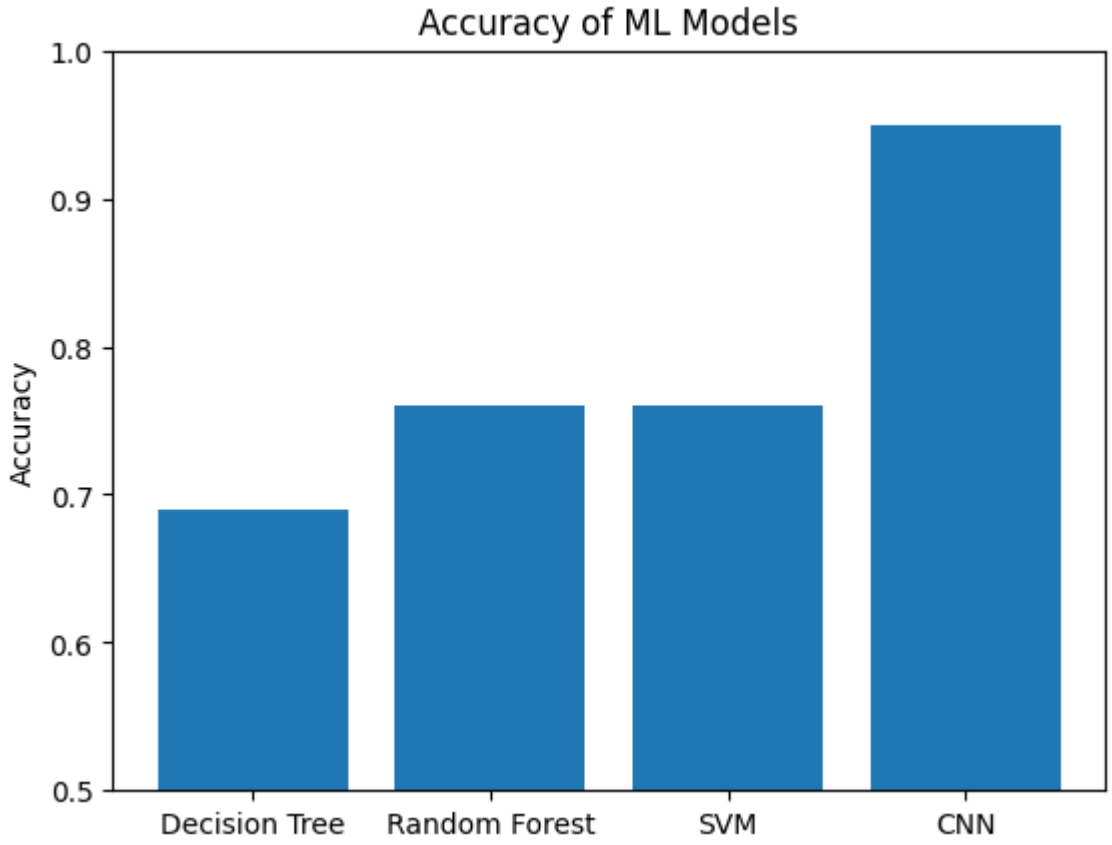




In [49]:

```
decision_tree=0.69
random_forest=0.76
svm=0.76
cnn=0.95
```

In [53]:
```
models = ['Decision Tree', 'Random Forest', 'SVM', 'CNN']
accuracies = [decision_tree, random_forest, svm, cnn]

plt.bar(models, accuracies)
plt.ylim([0.5, 1.0])
plt.ylabel('Accuracy')
plt.title('Accuracy of ML Models')


plt.show()
```



In [72]:
```
from tensorflow.keras.preprocessing import image
img_path = 'C:/Users/KIIT/Downloads/Telegram Desktop/00a7c269-3476-4d25-b744-44d6353cd921___GCREC_Bact.Sp
img = image.load_img(img_path, target_size=(256, 256))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = img_array/255.0

prediction = CNN.predict(img_array)
predicted_class = np.argmax(prediction)
classes[predicted_class]
```

```
1/1 [==============================] - 0s 172ms/step
```

Out[72]: 'Bacterial_spot'

In [67]:
```
from tensorflow.keras.models import load_model, save_model

# assuming you have a model object named "model"
save_model(CNN, 'C:/Users/KIIT/Desktop/DATASET/model/model1.h5')
```

In [ ]:
```

```