

Green Guardian

Jorge Muñoz Zanón, Oliver Lloyd, Nicolò Baldi, Carmen Robres de Veciana

Description

Green Guardian: Your Go-To Plant Care Companion

Green Guardian is an artificial intelligence designed to revolutionize plant care by leveraging image recognition technology. The main objective of this AI is to answer the main question: “Is my plant healthy?”. By analyzing pictures of plants, the AI provides users with detailed information on pest infestations, diseases, leaf conditions, flowering patterns, and soil conditions. It taps into diverse datasets, encompassing a wide array of plant images, classifications for soil textures, and information on plant illnesses, to deliver comprehensive insights.

This intelligent system not only evaluates the overall health of the plant but also offers practical guidance on tailored care routines. Green Guardian identifies potential issues in real-time and empowers users with precise instructions on how to address specific concerns. Its functionality extends beyond mere detection, aiming to educate users on proactive measures for maintaining the well-being of their plants. In essence, Green Guardian acts as a knowledgeable companion, using visual cues to provide actionable insights, fostering a greener and healthier environment for plants under its watchful eye.

How Does Green Guardian Work?

1. Upload a Plant Photo: Just take a picture with your smartphone or camera and upload it to Green Guardian.

2. AI-Powered Analysis: The advanced AI processes your photo, pinpointing crucial features and determining your plant's overall health.

3. **Customized Care Tips:** Get immediate, personalized advice on watering, feeding, pruning, and pest management, tailored to your plant's unique needs.

4. **Track Your Plant's Growth:** Monitor your plant's development over time. Green Guardian keeps a record of your plant's history and shows you how effective your care has been.

Bias and ethics issues

Addressing AI bias and ethics in the context of Green Guardian, an AI-driven plant care tool, involves a nuanced understanding of several interconnected issues. **The effectiveness of Green Guardian largely depends on the diversity and comprehensiveness of its training data.** If this data primarily includes plants from specific regions, the tool might underperform for species from other areas, leading to suboptimal care recommendations. This not only affects the accuracy of the tool but also raises concerns about equitable access and representation.

Moreover, there's a potential risk of users **becoming overly dependent** on the AI for plant care, which could erode traditional gardening skills and knowledge. This dependence becomes especially critical in commercial settings like agriculture, where AI-driven decisions could have larger ecological and economic impacts.

Privacy is another significant concern. The user data that Green Guardian collects, such as plant images and possibly user locations, needs to be handled with strict privacy controls to prevent misuse or unauthorized sharing.

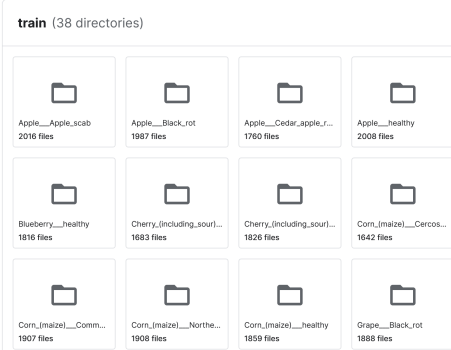
In terms of responsibility, the tool's developers bear a significant burden to ensure **the accuracy of their AI.** Misguided advice due to AI errors can lead to unintended consequences, including plant death or ecological imbalances, which is particularly significant in commercial or ecological contexts. This also relates to the fact that this tool **should be transparent** about how it reaches its conclusions and recommendations to empower users with informed decision-making.

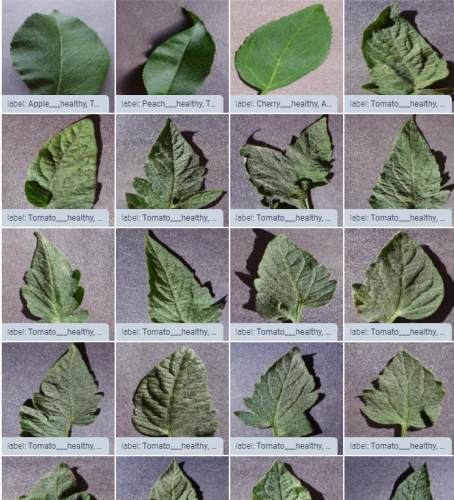

In essence, while Green Guardian presents a forward step in leveraging AI for plant care, it brings with it a spectrum of ethical and bias-related challenges that need careful consideration and management to ensure it is beneficial, fair, and sustainable in the long term.

Information of at least one dataset related to your task/s or a description of the process to create one

We combined a series of datasets to train the model so that we could accurately identify plants as well as the diseases that were affecting them. Each of these datasets was a large collection of images of different types of plant leaves affected by various diseases as well as healthy versions of the same plants (to allow image classification models to differentiate between healthy and diseased plants).

We used a total of 3 different datasets to help train the model:

Name	Picture	Link	Description
<p>New Plant Diseases Dataset</p>		<p>https://www.kaggle.com/datasets/vipooooo/new-plant-diseases-dataset</p>	<p>This dataset consists of about 87K rgb images of healthy and diseased crop leaves which are categorized into 38 different classes e.g. Apple Tree leaves, Potato leaves, Tomato plant leaves etc.</p>

<p>Plant Village</p>		<p>https://paperswithcode.com/dataset/plantvillage</p>	<p>The PlantVillage dataset consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease.</p>
<p>Plant disease recognition dataset</p>		<p>https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset</p>	<p>This dataset contains three labels, "Healthy", "Powdery", "Rust" referring to plant conditions. There is a total of 1530 images divided into train, test, and validation sets.</p>

Information of at least one API/tool/neural network related to your task/s

We used two different neural networks via API to create our plant care program “Green Guardian”. The first of these neural networks consists of an image classification network called “*nohamoamary/nabtah-plant-disease*” This model we used to correctly identify the type of plant presented in the image uploaded to it, as well as identify whether or not it is affected by some type of disease.

The second is a large language model from Meta known as *llama-2-70b-chat* which is a generative text model. This model was fed an input prompt that contained the identified plant

and plant disease, along with a question asking what should be done to help cure the plant. The input print string is as follows:

```
prompt = "My " + plant_name + " plant is getting " + disease_name + ",  
what should I do?"
```

These two neural networks were combined to create the final model that can correctly identify a plant and its illness and then recommend a series of treatments to cure the plant.

Simple code demo

Green Guardian

1. Setting Up

1.1. Installing Replicate

```
!pip install replicate
```

1.2. Inserting the API Token

```
import replicate  
api_token='r8_Ho0nIORDtcuUEYNgfUKzsgHaj1tV46u1Kulkt' #paste your token here  
client = replicate.Client(api_token=api_token)
```

1.3. Importing from Google Drive

```
#Potential Dataset: https://www.kaggle.com/code/sadikaljarif/plant-disease-classification-using-googlenet/input  
from google.colab import drive  
drive.mount('/content/drive')
```

2. Plant and Disease Recognition

2.1. Choose your Image



2.2. Playing the Image Recognition Model

```
#Image Recognition Model: nohamoamary/nabtah-plant-disease  
#Link: https://replicate.com/nohamoamary/nabtah-plant-disease  
  
import replicate  
output = client.run(  
    "nohamoamary/nabtah-plant-disease:33eabfb8b9664ec729b58d89d53e7ae8cd4e35979ebd5d27d2d1d95d88f7ee2",  
    input={"image": open("/content/drive/MyDrive/ExtendedIntelligences/AppleRust.jpg", "rb")} #paste your image path here  
)  
print(output)
```

2.3. Splitting the Output into Plant and Disease

```
split = output.split("___")  
plant_name = split[0]  
disease_name = split[1]  
print (plant_name)  
print (disease_name)  
  
Potato  
Early_blight
```

3. Disease Treatment

3.1. Playing the Language Model to suggest the treatment

#Language Model: llama-2-70b-chat

#Link: <https://replicate.com/meta/llama-2-70b-chat>

```
debug = False
top_k = 50
top_p = 1
prompt = "My " + plant_name + " plant is getting " + disease_name + ", what should I do?"
temp = 0.5
sys_pro = "You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any h
max_tok = 500
min_tok = -1
```

```
print(prompt + "\n")
```

```
import replicate
output_llama = client.run(
    "meta/llama-2-70b-chat:02e509c789964a7ea8736978a43525956ef40397be9033abf9fd2badfe68c9e3",
    input={"debug": debug,
          "top_k": top_k,
          "top_p": top_p,
          "prompt": prompt,
          "temperature": temp,
          "system_prompt": sys_pro,
          "max_new_tokens": max_tok,
          "min_new_tokens": min_tok}
)
```

```
for item in output_llama:
    print(item, end="")
```

Complete Code

```
#installreplicate
!pip install replicate
```

```
#import the token
import replicate
api_token='r8_Ho0nIORDtcuUEYNgfUKzsgHajitV4Gu1Kulkt' #paste your token here
client = replicate.Client(api_token=api_token)
```

```
#import drive folder
from google.colab import drive
drive.mount('/content/drive')
```

```
#image recognition to identify plant & disease
import replicate
output = client.run(
    "nohamamary/nabtah-plant-disease:33eabfb8b9664ec729b58d89d53e7ae8cd4e35979ebd5d27d22d1d95d88f7ee2",
    input={"image": open("/content/drive/MyDrive/ExtendedIntelligences/AppleRust.jpg", "rb")}) #paste your image path here
)
```

```
#split the output
split = output.split("___")
plant_name = split[0]
disease_name = split[1]
```

```
#language model to describe the treatment
```

```
debug = False
top_k = 50
top_p = 1
prompt = "\n" + plant_name + " plant is getting " + disease_name + ", what should I do?"
temp = 0.5
sys_pro = "You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any h
max_tok = 500
min_tok = -1
```

```
print(prompt + "\n")
```

```
import replicate
output_llama = client.run(
    "meta/llama-2-70b-chat:02e509c789964a7ea8736978a43525956ef40397be9033abf9fd2badfe68c9e3",
    input={"debug": debug,
          "top_k": top_k,
          "top_p": top_p,
          "prompt": prompt,
          "temperature": temp,
          "system_prompt": sys_pro,
          "max_new_tokens": max_tok,
          "min_new_tokens": min_tok}
)
```

```
for item in output_llama:
    print(item, end="")
```
