

Cassava Leaf Disease detection system using Azure AI , Container Apps and Cosmos DB

Problem Statement

We implement a **Cassava Leaf Disease detection system**.

Cassava is a rich, affordable source of carbohydrates. **It can provide more calories per acre of the crop than cereal grain crops, which makes it a very useful crop in developing nations.**

As the 2nd largest provider of carbohydrates in Africa, cassava is a **key food security crop grown by small-holder farmers** because it can withstand harsh conditions. At least 80% of small-holder farmer households in Sub-Saharan Africa grow cassava and viral diseases are major sources of poor yields.

We have taken 105 images for 5 categories, 4 leaf disease categories [**cbb, cbsd, cgm, cmd**] and a healthy category. The **Cassava Leaf Disease detection system** would help the farmers to detect the disease correctly and take preventive measures for the same.

A successful output for an **unseen cbb disease image** is shown with **99.93% probability for cbb diseased leaf**

Cassava Leaf Disease

Predictions

Prediction for train-cbb-378.jpg



cbb : 0.99933416

cbsd : 0.00061068207

cgm : 4.397634e-05

healthy : 6.0237385e-06

Successful predictions in table format

Cassava Leaf Disease

All Predictions

[Predict](#) [All Predictions](#)

All Predictions in Percentages

FileName

FILENAME	CBB	CBSD	CGM	CMD	HEALTHY
train-cbb-378.jpg	99.933	0.061	0.004	0.001	0.001
train-cbb-307.jpg	57.384	42.572	0.002	0.0	0.042
train-cbb-413.jpg	100.0	0.0	0.0	0.0	0.0
train-cgm-14.jpg	0.0	0.0	99.998	0.0	0.001
train-cgm-538.jpg	0.007	0.073	5.707	94.21	0.003
train-cgm-599.jpg	0.018	0.0	99.977	0.004	0.0
train-cgm-589.jpg	0.649	0.663	98.672	0.016	0.0
train-cgm-383.jpg	0.004	0.0	99.996	0.0	0.0
train-cgm-747.jpg	0.003	0.002	99.995	0.0	0.0
train-cmd-749.jpg	0.002	0.001	0.0	99.997	0.0
train-cmd-5.jpg	0.006	0.0	0.0	99.993	0.001

Searched predictions

Cassava Leaf Disease

All Predictions

[Predict](#) [All Predictions](#)

All Predictions in Percentages

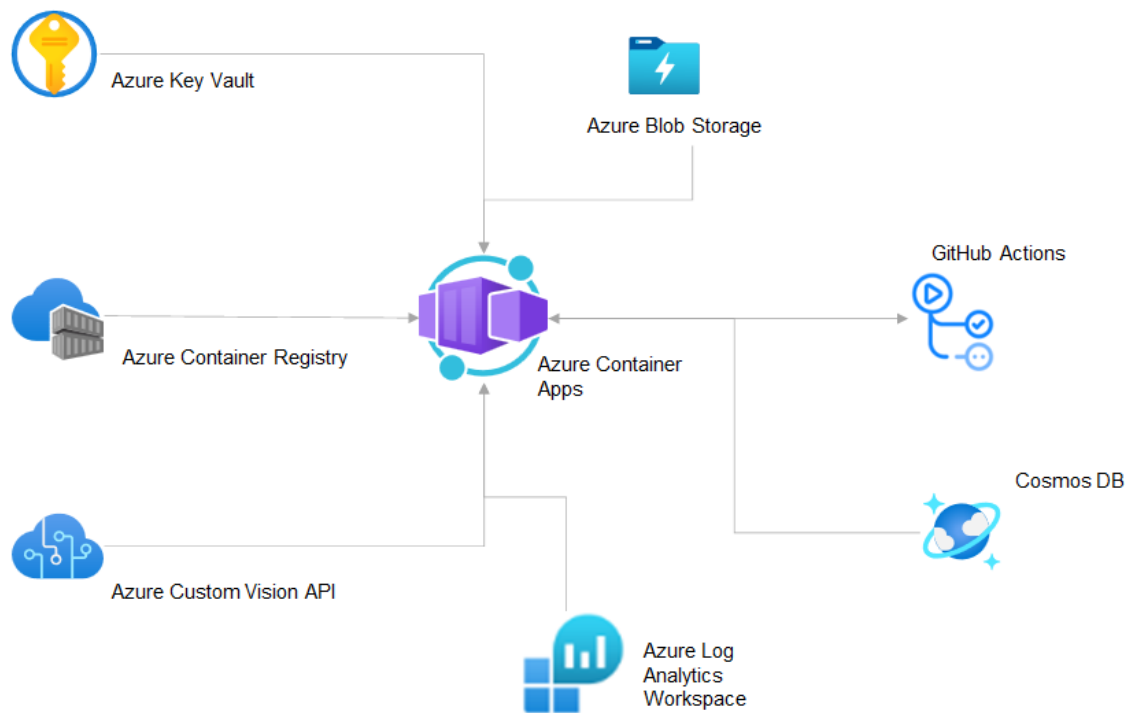
FileName

FILENAME	CBB	CBSD	CGM	CMD	HEALTHY
train-cbb-298.jpg	0.014	0.922	98.799	0.004	0.261
train-cbb-317.jpg	0.085	0.002	0.002	0.0	99.911
train-cbb-287.jpg	0.213	99.78	0.006	0.0	0.001
train-cbb-334.jpg	99.942	0.037	0.002	0.0	0.019
train-cbb-433.jpg	99.998	0.0	0.001	0.0	0.001
train-cbb-7.jpg	100.0	0.0	0.0	0.0	0.0
train-cbb-423.jpg	55.71	44.046	0.241	0.001	0.002
train-cbb-327.jpg	97.776	2.223	0.0	0.0	0.0
train-cbb-297.jpg	100.0	0.0	0.0	0.0	0.0
train-cbb-413.jpg	100.0	0.0	0.0	0.0	0.0
train-cbb-307.jpg	57.384	42.572	0.002	0.0	0.042
train-cbb-378.jpg	99.933	0.061	0.004	0.001	0.001

Architecture

The solution would be implemented as web application which is also mobile enabled. The farmers can access this application anywhere and can upload the images to detect the disease in the cassava leaves.

1. Flask is used as UI
2. The application is deployed as a **Container App**
3. **Azure Custom Vision API** used to detect the Cassava leaf diseases
4. **Azure Container Registry** used to store the container images
5. The secrets required for the solution are stored in **Azure Key Vault**
6. The Azure Container App uses **Managed Identity**
7. The integration of Azure Container App with **GitHub Actions** is used for CD[Continuous Deployment]
8. The images are stored in the **Azure Blob Storage**
9. The predictions obtained from the Azure Custom Vision API are stored in **Cosmos DB**.



Data Flow

- 1.** The image is uploaded into the Container App through the Flask UI
- 2.** The Container App
 - a.** uploads the image into Azure Blob Storage
 - b.** predicts the disease through the Azure Custom Vision API
 - c.** saves the filename and the predictions in Cosmos DB

Technical Details and Implementation of solution

The various components of the solution are

1. Custom Vision AI project for training the Cassava leaves
2. Deployment of the Container App
3. Configure the Container App for interaction with the
 - a. Storage Account
 - b. KeyVault
 - c. CosmosDB
 - d. GitHub Actions
4. Code walkthrough

Custom Vision AI Project

1. Create a Custom Vision AI project

Navigate to <https://www.customvision.ai/projects> to create a custom vision project.

We created a project with

Name - cassava

Project Type - **Classification**. Since we are classifying whether the image is having **cbb, cbsd, cgm, cmd and healthy** disease

Classification Type - **Multiclass**. There are 2 choices here, Multiclass and Multilabel. We choose Multiclass since the image is associated with only one class. A single image is not associated with multiple classes. If a single image was associated with multiple classes, then we had to choose the Classification type as Multilabel.

Create new project



Name*

Description

Resource*

[create new](#)[Manage Resource Permissions](#)

Project Types (i)

- Classification
- Object Detection

Classification Types (i)

- Multilabel (Multiple tags per image)
- Multiclass (Single tag per image)

Domains:

- General [A2]
- General [A1]
- General
- Food
- Landmarks
- Retail
- General (compact) [S1]
- General (compact)
- Food (compact)
- Landmarks (compact)
- Retail (compact)


Pick the domain closest to your scenario. Compact domains are lightweight models that can be exported to iOS/Android and other platforms. [Learn More](#)

2.Add Images

We upload the **cbb** images and also tag them. We upload the other images in the same way.

Image upload

○ Add Tags ● Uploading ● Summary



105 images will be added...


Add some tags to this batch of images...

My Tags

[Upload 105 files](#)

3.Train the images

We train the images by clicking the **Train button** in the portal

[Training Images](#) [Performance](#) [Predictions](#) [Train](#) [Quick Test](#) 

4.Training

Select **Quick Training** or **Advanced Training** for training the images

Choose Training Type



Training Types ⓘ

Quick Training

Advanced Training

Est. Minimum Budget: 1 hour

Train

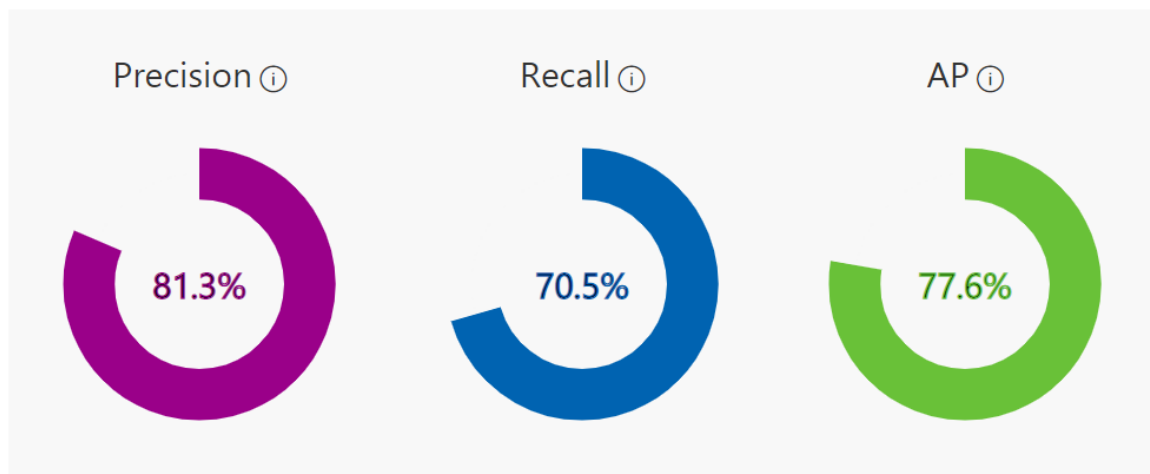
Choose Quick Training and the results are shown below

Iteration 1

Finished training on **10/22/2022, 11:51:16 AM** using **General [A2]** domain

Iteration id: **769c8daa-9cdb-4827-abfa-35292bffd0a7**

Classification type: **Multiclass (Single tag per image)**



We chose **Advanced Training** to train the images.

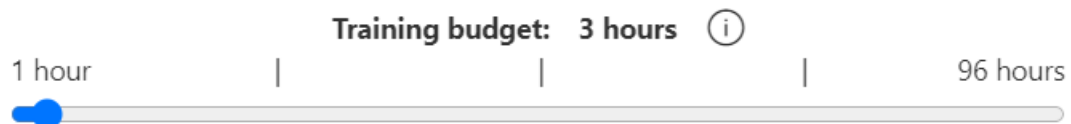
Choose Training Type



Training Types (i)

- Quick Training
- Advanced Training

In most cases, the more time you select the better the model will be. You're charged based on the compute time used to train your model, so choose your budget based on your need.



Send me an email notification after training completes

Email address

ambarish.ganguly@gmail.com

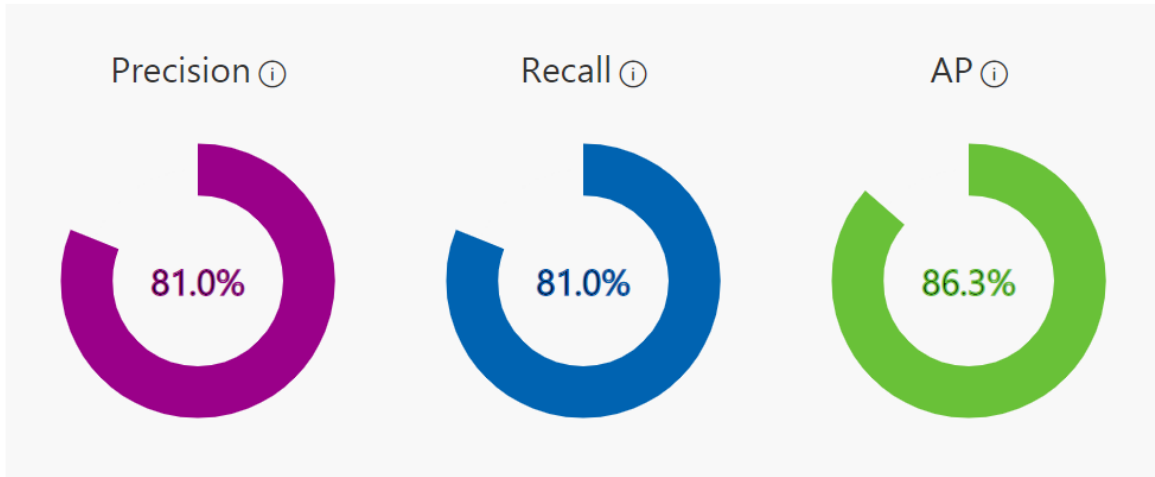
Est. Minimum Budget: 1 hour

Train

This is the **output of the Advanced Training with a training budget of 3 hours**. This provides much better accuracy compared to the Quick training as demonstrated in the picture below

Iteration 2

Finished training on **10/22/2022, 1:20:12 PM** using **General [A2]** domain
Iteration id: **51fdfb39-8ca6-4d60-b55a-02bb78df04f8**
Classification type: **Multiclass (Single tag per image)**



Performance Per Tag

Tag	Precision	^	Recall	A.P.	Image count
cmd	90.5%		90.5%	96.7%	105
cbds	81.8%		85.7%	96.9%	105
cbb	80.0%		57.1%	78.5%	105
cgm	77.3%		81.0%	86.2%	105
healthy	76.0%		90.5%	76.5%	105

5.Publish

Publish the model so that we can use the endpoint of the model for the prediction of unseen images.

Publish Model



We only support publishing to a prediction resource in the same region as the training resource the project resides in.

Please check if you have a prediction resource and if the prediction resource is in the same region as the training resource.

Model name

Prediction resource

Publish

Cancel

6. Project details

Azure Cognitive project which has the **project id**, the **published endpoint** is shown below. This will be used for predicting the unseen test images.

Project Settings

General

Project Name*

Project Id

Description

Settings

Resources:

cassava

Subscription: Visual Studio Enterprise Subscription

Resource Group: cassava

Resource Kind: All Cognitive Services

Key:

Endpoint:

Deployment of the Container App

Deploy the application as a Container App. **The detail instructions are in.**

<https://github.com/ambarishg/cassava-disease-detection/blob/master/docs/02-AzureContainerAppsSteps.md>

We turn the **Managed Identity [System Assigned] on** for the Container App. This setting will be used to access the Storage Account and the Key Vault

Home > cassavaflaskapp



cassavaflaskapp | Identity

Container App

Search

Diagnose and solve problems

Settings

Authentication

Secrets

Ingress

Continuous deployment

Custom domains

<<

System assigned

User assigned

A system assigned managed identity is restricted (RBAC). The managed identity is authenticated with



Save



Discard



Refresh



Status ⓘ

Off

On

Interaction of Container App with Storage Account

When the file is upload from the user interface, the container app uploads the file into the container of the storage account.

The Managed identity of the Container App is provided the role of the **Storage Blob Data Contributor** so that it can read, write , delete files from the container.

Home > cassavaflaskapp | Identity >

Azure role assignments

+ Add role assignment (Preview) Refresh

If this identity has role assignments that you don't have permission to read, they won't be shown.

Subscription *

Visual Studio Enterprise Subscription (6ea869be-bab3-4204-94c3-1fc677f7d2de)

Role

Resource Name

Storage Blob Data Contributor



stgcassava

Configure scaling in Container Apps

Create and deploy new revision ...

Container Scale

Scale rule setting

Control automatic scaling by setting the range of application replicas that'll be deployed in response to a trigger event. Use scale rules to determine the type of events that trigger scaling.

Min / max replicas * ⓘ 

Scale rule

+ Add

KeyVault name is stored as an environment variable secret in Container App



cassavaflaskapp | Secrets ...

Container App

Diagnose and solve problems

Settings

Authentication

Secrets

Ingress

Continuous deployment

<<

+ Add Refresh Send us your feedback

Secrets are key/value pairs than can be used to protect sen: store here will be valid across all your revisions. Note that c

Key ↑

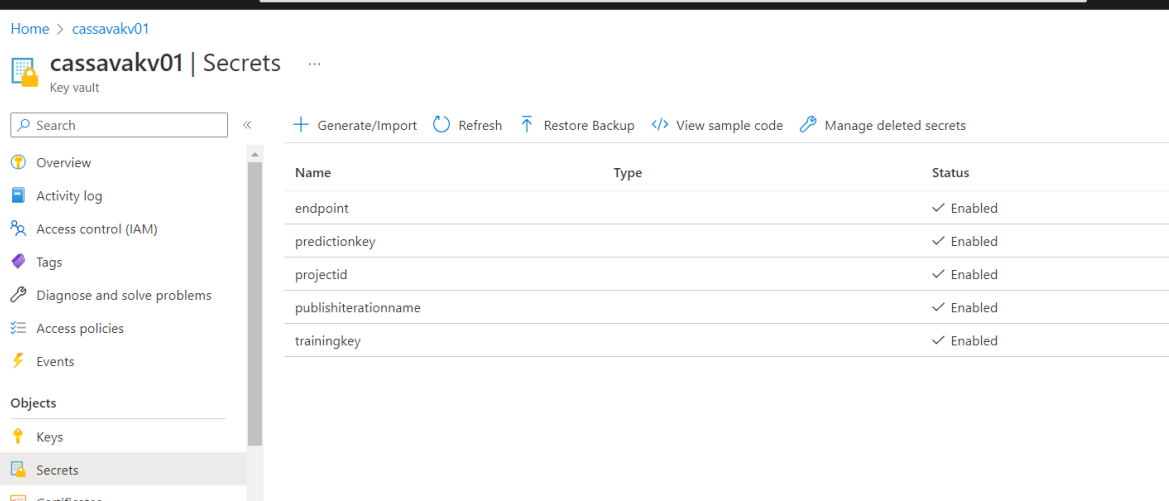
[cassavaflaskacrazurecrio-cassavaflaskacr](#)

[kvname-secret](#)

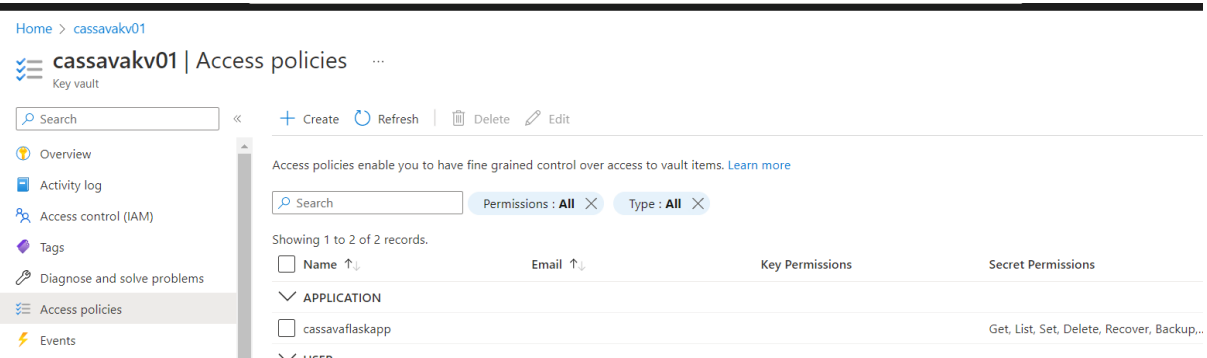
Interaction of Container App and the KeyVault

We put the following components as secrets in the KeyVault

Solution Component	Solution Element
Custom Vision API	prediction key , endpoint , project id and the publish iteration name
Storage Account	Storage account ,container
CosmosDB	CosmosDB account and the key



The Managed identity of Container App has access to read the secrets from the Key Vault using the Access Policies



Cosmos DB configuration

A Cosmos DB account [**agcosmos**] , database[**predictions**] and container[**predictions**] is created for storing the predictions. The partition used is /category which is currently "cassava". When the application is extended to other leaf diseases, it can be other leaves.

* Database id ⓘ

Create new Use existing

predictions

Share throughput across containers ⓘ

* Database throughput (autoscale) ⓘ

Autoscale Manual

Estimate your required RU/s with [capacity calculator](#).

Database Max RU/s ⓘ

1000 *

Your database throughput will automatically scale from **100 RU/s (10% of max RU/s) - 1000 RU/s** based on usage.

Estimated monthly cost (USD) ⓘ: **\$8.76 - \$87.60** (1 region, 100 - 1000 RU/s, \$0.00012/RU)

* Container id ⓘ

predictions

*** Partition key** ⓘ

For small workloads, the item ID is a suitable choice for the partition key.

/category

Unique keys ⓘ

+ Add unique key

Analytical store ⓘ

On Off

Azure Synapse Link is required for creating an analytical store container. Enable Synapse Link for this Cosmos DB account.

[Learn more](#)

Enable

> **Advanced**

OK

GitHub Actions

The Container App is also integrated with GitHub Actions for Continuous deployment

The screenshot shows the Azure Portal interface for a Container App named 'cassavaflaskapp'. The left sidebar contains navigation options: Overview, Access control (IAM), Tags, Diagnose and solve problems, Settings (Authentication, Secrets, Ingress, Continuous deployment, Custom domains, Dapr, Identity, Service Connector (preview)), and a search bar. The main content area is titled 'cassavaflaskapp | Continuous deployment'. It includes a 'Refresh' button, a 'Send us your feedback' link, and a notification: 'You're currently deploying to your app automatically using GitHub Actions. disconnect continuous deployment.' Below this is a section for 'GitHub settings' with the following details:

Setting	Value
Signed in as	ambarishg
Organization	ambarishg
Repository	cassava-disease-detection
Branch	master
Workflow file	.github/workflows

Code walkthrough using the Flask UI , Storage Account , KeyVault, Azure Custom Vision API and CosmosDB

We get the KeyVault name from the environment variable of the Container App

```
keyVaultName = os.environ["kvname"] #""
KVUri = f"https://{keyVaultName}.vault.azure.net"
```

We put the prediction key , endpoint , project id and the publish iteration name in the KeyVault. We access the secrets from the KeyVault using the Managed Identity.

```
credential = ManagedIdentityCredential()
client = SecretClient(vault_url=KVUri, credential=credential)
ENDPOINT = client.get_secret("endpoint").value
prediction_key = client.get_secret("predictionkey").value
prediction_resource_id = "paddy"
project_id = client.get_secret("projectid").value
publish_iteration_name = client.get_secret("publishiterationname").value
```

Code shows of how we upload the file into Azure Blob Storage

```
if uploaded_file is not None:
```

```

local_file_name = uploaded_file.name
bytes_data = uploaded_file.getvalue()
blob_service_client = BlobServiceClient(account_url,
credential=credential)
blob_client = blob_service_client.get_blob_client(container=container_name,
blob=local_file_name)
blob_client.upload_blob(bytes_data)

```

Code shows of how we are using to predict using the Azure Custom Vision API and also to store the predictions in Cosmos DB

```

prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(ENDPOINT, prediction_credentials)

results = predictor.classify_image(
    project_id, publish_iteration_name, bytes_data)

result_dict = {}
result_dict["category"] = "cassava"
result_dict["filename"] = local_file_name
result_dict["id"] = 'cassava_' + str(uuid.uuid4())

st.write("The cassava leaf image predictions are")
# Display the results.
for prediction in results.predictions:
    st.write("\t" + prediction.tag_name +
        ": {0:.2f}%".format(prediction.probability * 100))
    result_dict[prediction.tag_name] = round(prediction.probability * 100,3)
cosmosdbwithoutasync.create_item(result_dict)

```

Challenges in implementing the solution

The solution makes use of several Azure services such as Azure Container Apps, Azure Custom Vision, Cosmos DB, Key Vault , Managed Identity, Storage Account. Integrating it required considerable planning. The seamless integration between the Azure services helped to make the implementation easier.

Business Benefit

Cassava is a very important crop for the developing nations since it provides a rich source of carbohydrates. The solution presently would help the farmers to predict Cassava diseases and take preventive actions for increased crop yield. The solution is built using **scalable Azure services** and can be extended to other crops such as tomato, potatoes and many others.

GitHub Link

<https://github.com/ambarishg/cassava-disease-detection>