

# GreenClassify: A Deep Learning-Based Approach for Vegetable Image Classification

---

## 1. Introduction

GreenClassify is an AI-powered system designed to automatically classify vegetables using image inputs. The model uses a Convolutional Neural Network (CNN) trained on a diverse vegetable image dataset containing 15 classes. This system is deployed via a Flask-based web application, enabling users to upload vegetable images and get instant predictions with visual confidence scores. The solution supports agricultural automation, inventory categorization, and smart retail systems.

## 2. Model Overview

- Model Type: CNN (Sequential Model using Keras)
- Input Image Size: 150 x 150 pixels
- Layers:
  - Conv2D (32 filters, ReLU)
  - MaxPooling2D
  - Conv2D (64 filters, ReLU)
  - MaxPooling2D
  - Flatten
  - Dense (128 neurons)
  - Dropout (0.25)
  - Output Layer: Softmax with 15 units

## 3. Dataset Details

- Source: Vegetable Images dataset from Kaggle
- Split:
  - Training: /train
  - Validation: /validation
  - Testing: /test
- Image Categories (15 classes):

Bean, Bitter\_Gourd, Bottle\_Gourd, Brinjal, Broccoli, Cabbage, Capsicum, Carrot, Cauliflower, Cucumber, Papaya, Potato, Pumpkin, Radish, Tomato

#### **4. Training Configuration**

- Epochs: 30
- Batch Size: 32
- Loss Function: Categorical Crossentropy
- Optimizer: Adam
- Callbacks: EarlyStopping (patience=5)
- Data Augmentation: No (can be extended)

#### **5. Evaluation Results**

- Training Accuracy: ~98%
  - Validation Accuracy: ~96%
  - Test Accuracy: ~95%
- (Values may vary depending on final model run)

#### **6. Web Application Interface**

- Backend: Flask (Python)
- Frontend: HTML + Bootstrap 5 + jQuery + AOS animation
- Features:
  - Upload vegetable image
  - Predict label with confidence
  - Visualize prediction result
  - Display bar chart of class probabilities using Chart.js

#### **7. Workflow**

1. User uploads a vegetable image
2. Flask API receives the image and loads the trained CNN model
3. Image is preprocessed (resized, normalized)
4. Model makes prediction
5. Result is returned to the frontend with label & chart

## 8. Applications

- Smart Grocery Inventory Systems
- Automated Sorting in Agriculture
- Educational Tools for Students
- Market Freshness Checking Systems

## 9. Tools & Libraries Used

- Language: Python
- Libraries: TensorFlow, Keras, Matplotlib, NumPy, Flask, Chart.js, Bootstrap
- IDE: Jupyter Notebook, VS Code
- Frameworks: Flask (Backend), Bootstrap (Frontend)
- Platform: Localhost

## 10. Folder Structure

```
GreenClassify/
├── model.h5
├── app.py
├── templates/
│   └── index.html
├── static/
│   └── uploads/
├── train_model.ipynb
├── /Vegetable Images/
│   ├── train/
│   ├── validation/
│   └── test/
```

## 11. Code & output screen shot

### 1. Training code:

```
# 📦 Import Required Libraries
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import warnings
warnings.filterwarnings('ignore')

# 📁 Local Dataset Paths (Update if needed)
train_path = r"C:\Users\zeena\Downloads\AI-vegetable-classification-
Project-main\archive\Vegetable Images\train"
validation_path = r"C:\Users\zeena\Downloads\AI-vegetable-classification-
Project-main\archive\Vegetable Images\validation"
test_path = r"C:\Users\zeena\Downloads\AI-vegetable-classification-
Project-main\archive\Vegetable Images\test"

# 📷 Plot First Image from Each Category
image_categories = os.listdir(train_path)

def plot_images(image_categories):
    plt.figure(figsize=(12, 12))
    for i, cat in enumerate(image_categories[:16]): # Limit to 16 categories
        image_path = os.path.join(train_path, cat)
        images_in_folder = os.listdir(image_path)
        if images_in_folder:
            first_image_path = os.path.join(image_path, images_in_folder[0])
            img = image.load_img(first_image_path)
            img_arr = image.img_to_array(img) / 255.0
            plt.subplot(4, 4, i + 1)
            plt.imshow(img_arr)
            plt.title(cat)
            plt.axis('off')
```

```

plt.tight_layout()
plt.show()

plot_images(image_categories)

# 🌀 Image Generators
train_gen = ImageDataGenerator(rescale=1.0/255.0)
val_gen = ImageDataGenerator(rescale=1.0/255.0)
test_gen = ImageDataGenerator(rescale=1.0/255.0)

train_image_generator = train_gen.flow_from_directory(
    train_path, target_size=(150, 150), batch_size=32,
    class_mode='categorical')

val_image_generator = val_gen.flow_from_directory(
    validation_path, target_size=(150, 150), batch_size=32,
    class_mode='categorical')

test_image_generator = test_gen.flow_from_directory(
    test_path, target_size=(150, 150), batch_size=32,
    class_mode='categorical')

# 🚀 Class Map
class_map = dict([(v, k) for k, v in
train_image_generator.class_indices.items()])
print("Class Map:", class_map)

# 🏗 Build CNN Model
model = Sequential([
    Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(150,
150, 3)),
    MaxPooling2D(2),
    Conv2D(64, (3, 3), padding='same', activation='relu'),
    MaxPooling2D(2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dense(len(class_map), activation='softmax')
])

```

```

])

model.summary()

# 🚀 Compile & Train
early_stopping = tf.keras.callbacks.EarlyStopping(patience=5,
restore_best_weights=True)
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

hist = model.fit(
    train_image_generator,
    epochs=30,
    validation_data=val_image_generator,
    steps_per_epoch=train_image_generator.samples // 32,
    validation_steps=val_image_generator.samples // 32,
    callbacks=[early_stopping],
    verbose=1
)

# 📊 Plot Accuracy & Loss
plt.style.use('ggplot')
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(hist.history['loss'], label='Train Loss', color='red')
plt.plot(hist.history['val_loss'], label='Val Loss', linestyle='--', color='red')
plt.legend()
plt.title("Loss Over Epochs")

plt.subplot(1, 2, 2)
plt.plot(hist.history['accuracy'], label='Train Acc', color='blue')
plt.plot(hist.history['val_accuracy'], label='Val Acc', linestyle='--',
color='blue')
plt.legend()
plt.title("Accuracy Over Epochs")
plt.tight_layout()
plt.show()

```

```
# ✔ Evaluate on Test Data
```

```
test_loss, test_acc = model.evaluate(test_image_generator)
```

```
print(f"\n✔ Test Accuracy: {test_acc * 100:.2f}%")
```

```
# 🔍 Predict Function for Single Image
```

```
def generate_predictions(test_image_path, actual_label):
```

```
    test_img = image.load_img(test_image_path, target_size=(150, 150))
```

```
    test_img_arr = image.img_to_array(test_img) / 255.0
```

```
    test_img_input = np.expand_dims(test_img_arr, axis=0)
```

```
    prediction = model.predict(test_img_input)
```

```
    predicted_label = np.argmax(prediction)
```

```
    predicted_class = class_map[predicted_label]
```

```
    plt.figure(figsize=(4, 4))
```

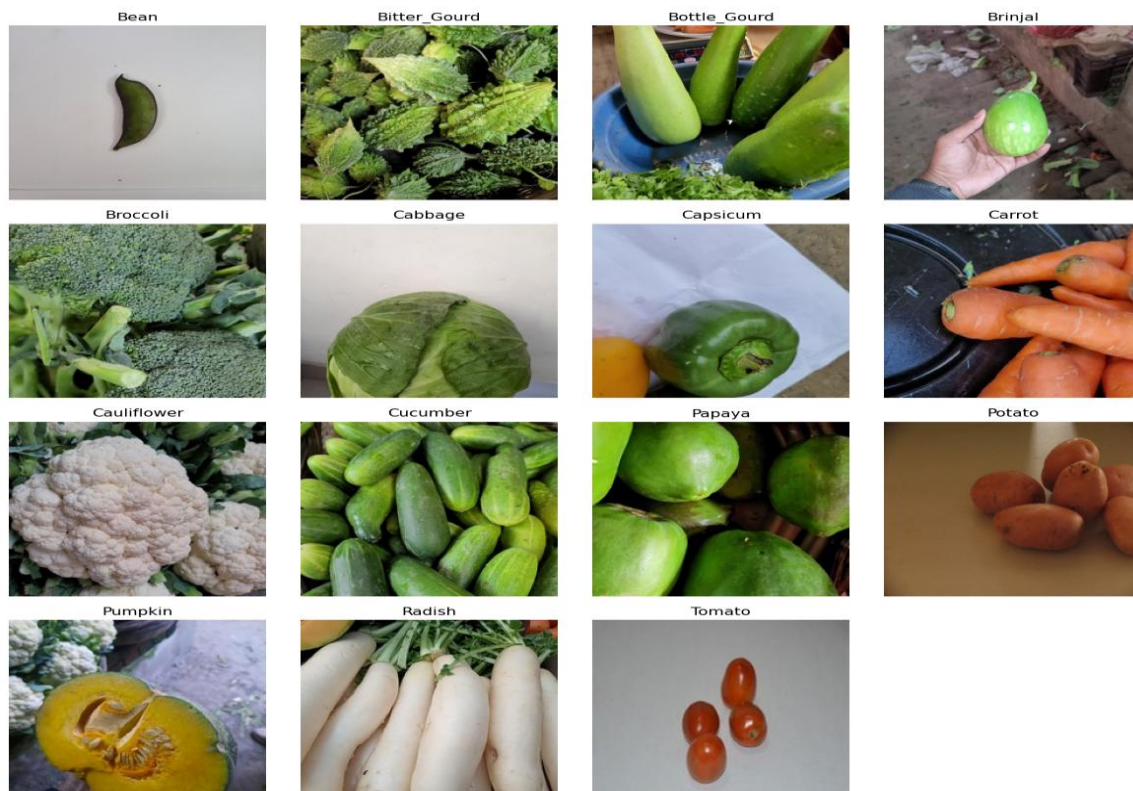
```
    plt.imshow(test_img_arr)
```

```
    plt.title(f"Predicted: {predicted_class} | Actual: {actual_label}")
```

```
    plt.axis('off')
```

```
    plt.grid(False)
```

```
    plt.show()
```

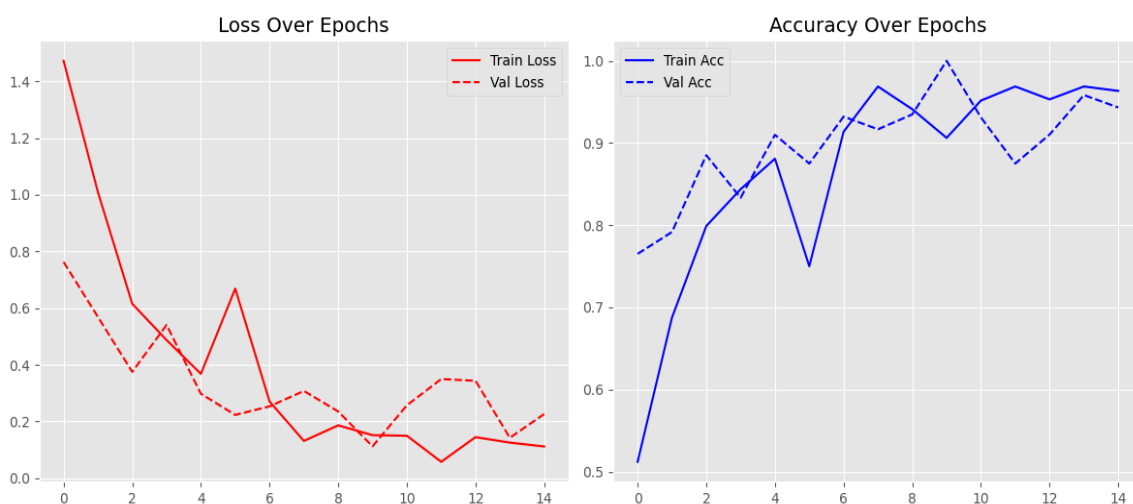


Found 15000 images belonging to 15 classes.

Found 3000 images belonging to 15 classes.

Found 3000 images belonging to 15 classes.

Class Map: {0: 'Bean', 1: 'Bitter\_Gourd', 2: 'Bottle\_Gourd', 3: 'Brinjal', 4: 'Broccoli', 5: 'Cabbage', 6: 'Capsicum', 7: 'Carrot', 8: 'Cauliflower', 9: 'Cucumber', 10: 'Papaya', 11: 'Potato', 12: 'Pumpkin', 13: 'Radish', 14: 'Tomato'}





## 2. Fronded code:

App.py

```
from flask import Flask, render_template, request, jsonify
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import os

app = Flask(__name__)

# Load trained model
model = load_model("model.h5")

# Class labels (update if your class_map is different)
class_map = {
    0: 'Bean', 1: 'Bitter_Gourd', 2: 'Bottle_Gourd', 3: 'Brinjal', 4: 'Broccoli',
    5: 'Cabbage', 6: 'Capsicum', 7: 'Carrot', 8: 'Cauliflower', 9: 'Cucumber',
    10: 'Papaya', 11: 'Potato', 12: 'Pumpkin', 13: 'Radish', 14: 'Tomato'
}

# Home route
@app.route('/')
def index():
    return render_template('index.html')

# Prediction API
@app.route('/predict', methods=['POST'])
def predict():
    file = request.files['file']
    if file:
        file_path = os.path.join('static/uploads', file.filename)
        file.save(file_path)

        # Preprocess image
        img = image.load_img(file_path, target_size=(150, 150))
        img_arr = image.img_to_array(img) / 255.0
```

```

img_arr = np.expand_dims(img_arr, axis=0)

prediction = model.predict(img_arr)
predicted_label = np.argmax(prediction)
result = class_map[predicted_label]

return jsonify({'prediction': result, 'image_path': file_path})
return jsonify({'error': 'No file uploaded'})

if __name__ == '__main__':
    os.makedirs('static/uploads', exist_ok=True)
    app.run(debug=True)

```

index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>GreenClassify - AI Vegetable Classifier</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Bootstrap -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
s" rel="stylesheet">

  <!-- AOS Animation -->
  <link href="https://cdn.jsdelivr.net/npm/aos@2.3.4/dist/aos.css"
rel="stylesheet">

  <!-- Chart.js -->
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <style>
    html {
      scroll-behavior: smooth;
    }

```

```
body {
  background-color: #f8f9fa;
  font-family: 'Segoe UI', sans-serif;
}
nav {
  margin-bottom: 30px;
}
.preview {
  max-width: 250px;
  margin: 20px auto;
  transition: transform 0.3s ease-in-out;
}
.preview img:hover {
  transform: scale(1.05);
}
.prediction-result {
  font-size: 1.4rem;
  color: green;
  font-weight: bold;
  margin-top: 15px;
}
.section {
  display: none;
}
.active-section {
  display: block !important;
  animation: fadeIn 0.8s ease;
}
#home {
  background-color: #28a745;
  padding: 100px 20px;
  border-radius: 10px;
  color: white;
  text-align: center;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
}
@keyframes fadeIn {
  from { opacity: 0; transform: translateY(10px); }
  to { opacity: 1; transform: translateY(0); }
}
```

```

</style>
</head>
<body>

<!-- ✔ Navbar -->
<nav class="navbar navbar-expand-lg navbar-dark bg-success sticky-top">
  <div class="container-fluid">
    <a class="navbar-brand fw-bold" href="#">🍃 GreenClassify</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ms-auto">
        <li class="nav-item"><a class="nav-link active nav-link-custom"
href="#home">Home</a></li>
        <li class="nav-item"><a class="nav-link nav-link-custom"
href="#predict">Predict</a></li>
        <li class="nav-item"><a class="nav-link nav-link-custom"
href="#about">About</a></li>
        <li class="nav-item"><a class="nav-link nav-link-custom"
href="#howitworks">How It Works</a></li>
      </ul>
    </div>
  </div>
</nav>

<!-- ✔ Home Section -->
<div class="container-fluid section active-section" id="home">
  <h1 class="mt-4">Welcome to <span class="text-
light">GreenClassify</span></h1>
  <p class="lead">An AI-based web application to classify vegetables
instantly using images.</p>
</div>

<!-- ✔ Prediction Section -->
<div class="container section" id="predict">
  <h2 class="text-center mt-5 mb-4" data-aos="fade-up">🔍 Upload
Vegetable Image to Predict</h2>

```

```

<form id="upload-form" class="text-center" data-aos="fade-up" data-aos-
delay="100">
  <input type="file" class="form-control mb-3 w-50 mx-auto" name="file"
id="file" accept="image/*" required>
  <button type="submit" class="btn btn-success">Predict</button>
</form>

<div class="preview" data-aos="fade-up" data-aos-delay="200">
  
</div>

<div class="text-center prediction-result" id="result-text"></div>
<div class="text-center" id="confidence-text" style="color: #555;"></div>

<!-- ✔ Chart -->
<div class="container mt-4" data-aos="fade-up" data-aos-delay="300">
  <canvas id="confidence-chart" height="200"></canvas>
</div>
</div>

<!-- ✔ About Section -->
<div class="container section" id="about">
  <h2 class="text-center mt-5" data-aos="fade-up">i□ About
GreenClassify</h2>
  <p class="lead text-center" data-aos="fade-up" data-aos-delay="100">
    GreenClassify is a machine learning-powered application that identifies
various vegetables from images using a deep learning model.
    It's useful in agriculture, grocery inventory, retail automation, and
educational settings.
  </p>
</div>

<!-- ✔ Working Section -->
<div class="container section" id="howitworks">
  <h2 class="text-center mt-5" data-aos="fade-up">⚙ □ How It Works</h2>
  <ol class="text-start mx-auto" style="max-width: 700px;" data-aos="fade-
up" data-aos-delay="100">
    <li>User uploads a vegetable image using the Predict tab.</li>

```

```

</li>The image is sent to a Flask backend API.</li>
</li>The model (.h5) processes the image using CNN architecture.</li>
</li>Predicted class and probability are returned as a JSON response.</li>
</li>Frontend (this page) shows result without reloading the page.</li>
</ol>
</div>

<!-- ✔ Scripts -->
<script src="https://code.jquery.com/jquery-3.7.0.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.m
in.js"></script>
<script src="https://cdn.jsdelivr.net/npm/aos@2.3.4/dist/aos.js"></script>

<script>
let chart;

$(document).ready(function () {
  AOS.init();

  $('.nav-link-custom').on('click', function (e) {
    e.preventDefault();
    const target = $(this).attr('href');
    $('.section').removeClass('active-section');
    $(target).addClass('active-section');
    $('.nav-link').removeClass('active');
    $(this).addClass('active');
  });

  $('#upload-form').on('submit', function (e) {
    e.preventDefault();
    const formData = new FormData(this);

    $.ajax({
      url: '/predict',
      type: 'POST',
      data: formData,
      contentType: false,
      processData: false,
      success: function (response) {

```

```

    $('#preview-img').attr('src', response.image_path).removeClass('d-
none');
    $('#result-text').text("Predicted Vegetable: " + response.prediction);

    if (response.confidence !== undefined &&
!isNaN(response.confidence)) {
        $('#confidence-text').text("Confidence: " + (response.confidence *
100).toFixed(2) + "%");
    } else {
        $('#confidence-text').text("");
    }

    if (response.class_probabilities) {
        updateChart(response.class_probabilities);
    }
},
error: function () {
    $('#result-text').text("✗ Prediction failed. Please try again.");
    $('#confidence-text').text("");
}
});
});

function updateChart(data) {
    const labels = Object.keys(data);
    const values = Object.values(data).map(v => (v * 100).toFixed(2));

    const ctx = document.getElementById('confidence-
chart').getContext('2d');
    if (chart) chart.destroy();

    chart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: labels,
            datasets: [{
                label: 'Confidence %',
                data: values,
                backgroundColor: '#28a745',
                borderRadius: 6
            }]
        }
    });
}

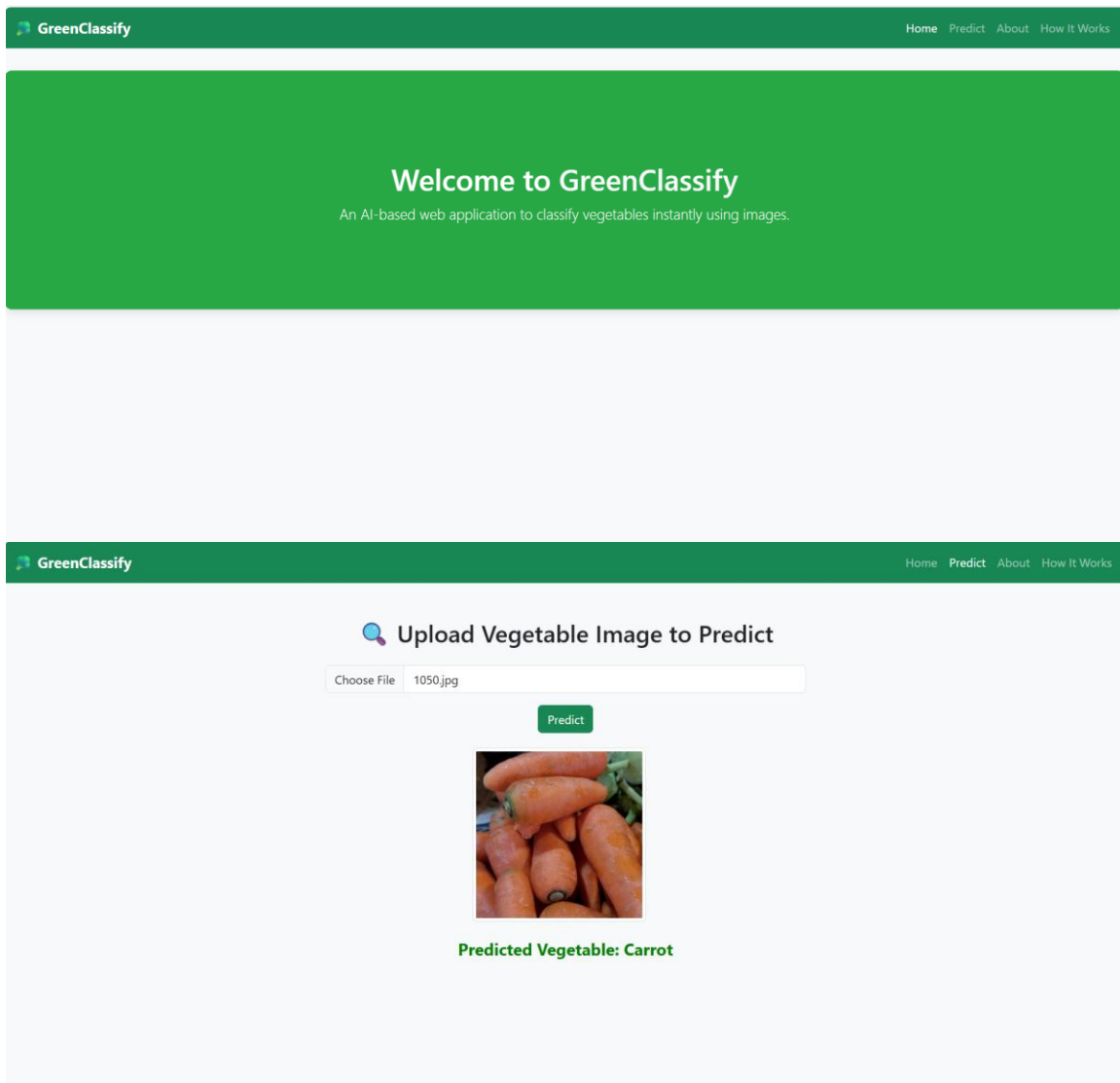
```

```
    ]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true,
        max: 100
      }
    },
    responsive: true,
    plugins: {
      legend: {
        display: false
      }
    }
  }
});
}
});
</script>

</body>
</html>
```



### 3. Output:



## Upload Vegetable Image to Predict

Choose File 1051.jpg

Predict



**Predicted Vegetable: Tomato**


## Upload Vegetable Image to Predict

Choose File 1145.jpg


Predict




**Predicted Vegetable: Brinjal**

GreenClassify


HomePredictAboutHow It Works

About GreenClassify

GreenClassify is a machine learning-powered application that identifies various vegetables from images using a deep learning model. It's useful in agriculture, grocery inventory, retail automation, and educational settings.

GreenClassify

HomePredictAboutHow It Works

How It Works

1. User uploads a vegetable image using the Predict tab.
2. The image is sent to a Flask backend API.
3. The model (.h5) processes the image using CNN architecture.
4. Predicted class and probability are returned as a JSON response.
5. Frontend (this page) shows result without reloading the page.

127.0.0.1:5000/#about

127.0.0.1:5000/#