BUSITEMA UNIVERSITY
Pursuing Excellence

FACULTY OF ENGINEERING AND TECHNOLOGY

ASSIGNMENT REPORT FOR 5.2 ON A PROJECT USING MATLAB CODE TO DESIGN AND SIMULATE THE YIELD ESTIMATION IN CROP PRODUCTION.

PRESENTED TO
THE COMPUTER PROGRAMMING COURSE LECTURER
MR. MASERUKA BENDICTO

By GROUP 17

GROUP SEVENTEEN MEMBERS

	NAME	REG NUMBER	PROGRAM
1	AHAISIBWE CHRISTOPHER	BU/UP/2024/0824	AMI
2	GIFT EMMANUEL	BU/UP/2024/3250	WAR
3	ADWONG CALEB	BU/UP/2024/3813	MEB
4	OWINO POSH IAN	BU/UP/2024/1067	WAR
5	AHEREZA FAITH	BU/UG/2024/2673	APE
6	MUWUBYA WYCLIFFE	BU/UP/2024/1045	WAR
7	NABUKEERA ANNET RIONAH	BU/UG/2024/2676	AMI
8	AKELLO BARBRA	BU/UP/2024/1001	WAR
9	ATATI EDWINE	BU/UP/2024/3730	AMI
10	OPOKA VINCENT	BU/UG/2024/1058	AMI

ACKNOWLEDGEMENT

We thank the Almighty god for the continued guidance and protection of our lives and highly appreciate our lecturer Mr. Benedicto Maseruka for always guiding us.

ABSTRACT

This report presents a comprehensive study on crop yield estimation using MATLAB-based data analysis and modeling techniques. The project aims to develop a reliable system for predicting agricultural output by analyzing key environmental and crop parameters such as rainfall, temperature, soil moisture, and fertilizer usage. MATLAB was employed for data preprocessing, feature extraction, and model development using both statistical and machine learning approaches. The implemented algorithms estimate yield patterns with improved accuracy by correlating real-world datasets with historical yield trends. Results indicate that the proposed model effectively captures the relationship between climatic and soil factors influencing productivity, providing an efficient tool for farmers, researchers, and policymakers to make informed agricultural decisions. The study highlights the potential of MATLAB in precision agriculture through automation, visualization, and performance optimization of predictive models.

DEDICATION

We dedicate this report to all group 17 members for their continued dedication to work tirelessly for the success of the whole group and to our lecturer for his continued support and guidance.

DECLARATION

We hereby declare that the information in this report was composed by us and has never been presented by anyone for any academic reward

	NAME	REG NUMBER	PROGRAM	SIGN
1	AHAISIBWE CHRISTOPHER	BU/UP/2024/0824	AMI	
2	GIFT EMMANUEL	BU/X/2024/3250	WAR	
3	NABUKEERA ANNET RIONAH	BU/UG/2024/2676	AMI	
4	ATATI EDWINE	BU/UP/2024/3730	AMI	
5	OKELLO BARBRA	BU/UP/2024/1001	WAR	
E6	AHEREZA FAITH	BU/UP/2024/2673	APE	
7	OWINO POSH IAN	BU/UP/2024/1067	WAR	
8	OPOKA VINCENT	BU/UG/2024/2675	AMI	
9	MUWUBYA WYCLIFFE	BU/UP/2024/1045	WAR	
10	ADWONG CALEB	BU/UP/2024/3813	MEB	

APPROVAL

We are presenting this report, which was successfully written and produced, under our efforts as Group 17 giving details of the assignment carried out.

Name………………………………………………………………………………………

Date……………………………………………………………………………………..

Signature……………………………………………………………………………..

TABLE OF CONTENTS

CHAPTER ONE

Problem statement

Accurate prediction of crop yield is a challenge in modern agriculture. Traditional methods often rely on manual observations or historical averages, which are prone to errors and cannot efficiently account for the dynamic effects of climatic conditions, soil quality, and agricultural practices. The inability to accurately predict yields can lead to inefficient resource allocation, economic losses, and food insecurity. Therefore, there is a need for a reliable, data-driven system that can analyze multiple factors affecting crop productivity and provide precise yield predictions in a timely manner.

Objectives:

1. To collect and preprocess relevant agricultural data, including soil characteristics, rainfall, temperature, and fertilizer usage.
2. To implement feature extraction and data analysis techniques using MATLAB for identifying key factors influencing crop yield.
3. To design and test predictive models using statistical and machine learning approaches for accurate yield estimation.
4. To evaluate the performance of the developed system and compare predicted yields with historical data.
5. To provide recommendations for precision agriculture practices based on the findings.

CHAPTER TWO: METHODOLOGY

At the beginning a meeting was held with all group members and discussed away forward. One of our group members went ahead to explain to us and teach us how we can develop the codes using matlab. Another meeting was held where the codes were developed.

DATASET

A csv file dataset was obtained from Kaggle on crop yield estimation giving us the parameters used for training our module using its conditions and crop varieties

CODES

2.1 CROP MODEL

It serves as a template for all crop yield estimation models and demonstrates OOP principles such as abstraction, encapsulation and polymorphism

```
classdef (Abstract) CropModel
    % CROPMODEL: Abstract base class for all crop yield estimation models.

    properties (SetAccess = private)
        CropType % Name of the crop the model is trained for.
    end

    properties (Access = protected)
        % Protected property for internal model structure (Encapsulation).
        ModelStructure
    end

    methods
        function obj = CropModel(cropType)
            % Constructor
            obj.CropType = cropType;
        end

        function displaySummary(obj)
            % Standard method to display model details.
            disp(['Model Type: ', class(obj)]);
            disp(['Crop: ', obj.CropType]);
        end
    end

    methods (Abstract)
        % Abstraction & Polymorphism: Methods must be implemented by concrete
subclasses.
```

```
        % Abstract method: Trains the model on features X and target Y.
        obj = trainModel(obj, X, Y)

        % Abstract method: Predicts yield for new features X_new.
        predictedYield = predictYield(obj, X_new)
    end
end
```

2.2 LINEAR REGRESSION CROP MODEL

It extends the abstract crop model to implement a linear regression-based crop yield estimation model and demonstrates key OOP concepts such as inheritance, polymorphism, encapsulation, abstraction, and method overriding. It provides a ready-to-use linear regression model for estimating crop yields, enforces consistent interface from crop model, making it easy to integrate with other crop models and allows training, prediction, and inspection of the model in a structured, maintainable way.

```
classdef LinearRegressionCropModel < CropModel
    % LINEARREGRESSIONCROPMODEL: A concrete yield model using linear regression.

    properties
        Coefficients % Model coefficients (Encapsulation)
        Rsquared     % R-squared value
    end

    methods
        function obj = LinearRegressionCropModel(cropType)
            % Calls the superclass constructor (Inheritance)
            obj = obj@CropModel(cropType);
        end

        function obj = trainModel(obj, X, Y)
            % Polymorphism: Concrete implementation of trainModel for Linear
Regression.
            disp(['Training Linear Regression Model for ', obj.CropType, '...']);

            % FIX: Combine the feature table (X) and response vector (Y) into a
single table (T).
            % This preserves the predictor column names for the model structure.
            T = [X, table(Y, 'VariableNames', {'TargetYield'})];

            % Matlab's fitlm function: Use the specified formula with named
variables.
            LM = fitlm(T, 'TargetYield ~ average_rain_fall_mm_per_year +
pesticides_tonnes + avg_temp');

            % Store results in the inherited/own properties (Encapsulation)
            obj.ModelStructure = LM;
            obj.Coefficients = LM.Coefficients.Estimate;
            obj.Rsquared = LM.Rsquared.Ordinary;
            disp('Training complete.');
        end

        function predictedYield = predictYield(obj, X_new)
            % Polymorphism: Concrete implementation of predictYield.

            if isempty(obj.ModelStructure)
                error('LinearRegressionCropModel:NotTrained', 'Model must be trained 
first.');
            end

            % X_new is already a table with the correct column names,
            % which now match the names stored in obj.ModelStructure.
            predictedYield = predict(obj.ModelStructure, X_new);
        end

        function displaySummary(obj)
            % Overriding and extending the inherited displaySummary method.
            displaySummary@CropModel(obj); % Call parent method (Inheritance)

            disp(' Model Statistics ');
            disp(['R-squared: ', num2str(obj.Rsquared)]);
            disp('Coefficients (Intercept, Rainfall, Pesticides, Temperature):');
            disp(obj.Coefficients');
        end
    end
end
```

2.3 YEILD DATA

It handles data loading, cleaning, preprocessing, and retrieval for crop yield estimation workflows.

```
classdef YieldData
    % YIELDDATA: Handles the loading, cleaning, and preparation of yield estimation 
data.

    properties (Access = private)
        % Encapsulation: 'Data' is protected from direct external modification.
        Data table
    end

    methods
        function obj = YieldData(fileName)
            % Constructor: Loads the data upon object creation.
            if nargin > 0
                obj = obj.loadData(fileName);
            end
        end

        function obj = loadData(obj, fileName)
            % Loads the dataset (assumed to be yield_df.csv, which is pre-merged).
            try
                disp(['Loading data from: ', fileName]);
                obj.Data = readtable(fileName);
            catch ME
                error('YieldData:LoadError', 'Could not load the data file: %s',
ME.message);
            end
        end

        function obj = preprocessData(obj)
            % Data Cleaning and Feature Engineering (Abstraction of cleaning logic).
            disp('Preprocessing data...');

            % 1. Remove rows with missing values (NaNs in 'hg/ha_yield' or features)
            obj.Data = rmmissing(obj.Data);

            disp('Data preprocessing complete.');
        end

        function [X, Y, CropList] = getTrainingData(obj)
            % Provides the full preprocessed features (X) and target (Y) variables.

            if isempty(obj.Data)
                error('YieldData:NoData', 'Data has not been loaded or is empty.');
            end

            % Define features (X) and target (Y) based on yield_df.csv columns.
            FeatureCols = {'average_rain_fall_mm_per_year', 'pesticides_tonnes',
'avg_temp'};
            TargetCol = 'hg_ha_yield';

            % Abstraction: Only returns the necessary matrices, hiding table
complexity.
            X = obj.Data(:, FeatureCols);
            Y = obj.Data{:, TargetCol};
            CropList = unique(obj.Data.Item);
        end

        % FIX: NEW PUBLIC METHOD to safely filter the data
        function [X_crop, Y_crop] = filterDataByCrop(obj, cropName)
            % Returns the features and target variables filtered for a specific crop.
            if isempty(obj.Data)
                error('YieldData:NoData', 'Data has not been loaded or is empty.');
            end

            cropIndex = strcmp(obj.Data.Item, cropName);

            % Define features (X) and target (Y)
            FeatureCols = {'average_rain_fall_mm_per_year', 'pesticides_tonnes',
'avg_temp'};
            TargetCol = 'hg_ha_yield';

            X_crop = obj.Data(cropIndex, FeatureCols);
            Y_crop = obj.Data{cropIndex, TargetCol};
        end
```

```
    end
end
```

CROP YIELD ESTIMATOR APP USER INTERFACE

```
classdef CropYieldEstimatorApp < handle
    % CROPPREDICTORAPP: A MATLAB GUI for crop yield prediction with visualization.

    properties
        Fig                % Main figure object
        YieldDataObj       % Handle to the loaded and preprocessed YieldData object
        CropModelObj       % Handle to the trained LinearRegressionCropModel object

        % UI Components
        CropDropdown
        TrainButton
        SummaryText
        PlotAxes           % Axes for plotting
        RainfallEdit
        PesticidesEdit
        TempEdit
        PredictButton
        ResultText
    end

    methods
        % Constructor (App Startup)
        function app = CropYieldEstimatorApp()
            % 1. Load Data and Initialize Backend
            app.YieldDataObj = YieldData('yield_df.csv');
            app.YieldDataObj = app.YieldDataObj.preprocessData();

            % 2. Build the User Interface
            app.createUI();

            % 3. Populate Crop Dropdown
            [~, ~, cropList] = app.YieldDataObj.getTrainingData();
            app.CropDropdown.Items = [{'Select Crop'}; cropList];

            % 4. Set Initial UI State
            app.TrainButton.Enable = 'off';
            app.PredictButton.Enable = 'off';
        end

        % UI Creation Function
        function createUI(app)
            % Adjust figure size to accommodate the plot
            app.Fig = uifigure('Name', 'Crop Yield Predictor', 'Position', [100 100 600 550]);

            % --- 1. Model Training & Diagnostics Panel (Adjusted Position/Size) ---
            TrainPanel = uipanel(app.Fig, 'Title', '1. Model Training & Diagnostics', 'Position',
[20 250 560 300]);
```

```
            % Crop Selection and Train Button (Top Row)
            uilabel(TrainPanel, 'Position', [20 250 100 22], 'Text', 'Select Crop:');
            app.CropDropdown = uidropdown(TrainPanel, ...
                'Position', [120 250 180 22], ...
                'Items', {'Loading...'}, ...
                'ValueChangedFcn', @(src, event) app.CropDropdownValueChanged(src, event));

            app.TrainButton = uibutton(TrainPanel, 'push', ...
                'Position', [320 250 180 22], ...
                'Text', 'Train Model', ...
                'ButtonPushedFcn', @(src, event) app.TrainModelButtonPushed(src, event));

            % Summary Text (Below Top Row)
            uilabel(TrainPanel, 'Position', [20 220 100 22], 'Text', 'R-squared:');
            app.SummaryText = uilabel(TrainPanel, 'Position', [120 220 380 22], 'Text', 'No model
trained.');

            % UIAxes for Plotting - **FIXED:** Removed invalid properties from constructor
            app.PlotAxes = uiaxes(TrainPanel, 'Position', [20 10 520 200]);

            % --- 2. Predict Yield Panel (Adjusted Position/Size) ---
            PredictPanel = uipanel(app.Fig, 'Title', '2. Predict Yield (hg/ha)', 'Position', [20
20 560 210]);

            % Input Fields
            uilabel(PredictPanel, 'Position', [20 140 150 22], 'Text', 'Rainfall (mm/year):');
            app.RainfallEdit = uieditfield(PredictPanel, 'numeric', 'Position', [180 140 100
22]);

            uilabel(PredictPanel, 'Position', [20 110 150 22], 'Text', 'Pesticides (tonnes):');
            app.PesticidesEdit = uieditfield(PredictPanel, 'numeric', 'Position', [180 110 100
22]);

            uilabel(PredictPanel, 'Position', [20 80 150 22], 'Text', 'Avg Temp (Celsius):');
            app.TempEdit = uieditfield(PredictPanel, 'numeric', 'Position', [180 80 100 22]);

            app.PredictButton = uibutton(PredictPanel, 'push', ...
                'Position', [350 110 150 50], ...
                'Text', 'Predict Yield', ...
                'ButtonPushedFcn', @(src, event) app.PredictYieldButtonPushed(src, event));

            % Result Display
            uilabel(PredictPanel, 'Position', [20 30 150 22], 'Text', 'Predicted Yield:');
            app.ResultText = uilabel(PredictPanel, 'Position', [180 30 300 22], 'Text', '---');
        end

        % Callbacks

        function CropDropdownValueChanged(app, src, ~)
            % Reset status when crop selection changes
            if strcmp(src.Value, 'Select Crop')
                app.TrainButton.Enable = 'off';
            else
                app.TrainButton.Enable = 'on';
            end
            app.SummaryText.Text = 'No model trained.';
            app.ResultText.Text = '---';
            app.PredictButton.Enable = 'off';
            cla(app.PlotAxes); % Clear the plot axes
        end

        function TrainModelButtonPushed(app, ~, ~)
            cropName = app.CropDropdown.Value;

            try
                % 1. Get filtered data for the selected crop
                [X_crop, Y_crop] = app.YieldDataObj.filterDataByCrop(cropName);

                % 2. Initialize and Train the Linear Regression Model
                app.CropModelObj = LinearRegressionCropModel(cropName);
                app.CropModelObj = app.CropModelObj.trainModel(X_crop, Y_crop);

                % 3. Update UI
                rSquared = app.CropModelObj.Rsquared;
                app.SummaryText.Text = sprintf('%.4f', rSquared);
                app.PredictButton.Enable = 'on';

                % 4. Plotting Logic: Rainfall vs. Yield with Regression Line

                rainFeatureCol = 'average_rain_fall_mm_per_year';

                % Scatter plot of actual data
                scatter(app.PlotAxes, X_crop{:, rainFeatureCol}, Y_crop, 15, 'filled', ...
                    'MarkerFaceColor', [0.4 0.7 1.0], 'DisplayName', 'Actual Data');
                hold(app.PlotAxes, 'on');

                % Calculate prediction line range
                x_min = min(X_crop{:, rainFeatureCol});
                x_max = max(X_crop{:, rainFeatureCol});
                x_range = linspace(x_min, x_max, 100)';

                % Set other features (Pesticides and Temp) to their mean for plotting context
                pesticide_mean = mean(X_crop.pesticides_tonnes);
                temp_mean = mean(X_crop.avg_temp);

                % Create a prediction table for the range
                X_pred = table(x_range, repmat(pesticide_mean, 100, 1), repmat(temp_mean, 100,
1), ...
                    'VariableNames', {'average_rain_fall_mm_per_year', 'pesticides_tonnes',
'avg_temp'});

                % Predict the yield across the rainfall range
                Y_pred = app.CropModelObj.predictYield(X_pred);

                % Plot the regression line
                plot(app.PlotAxes, x_range, Y_pred, 'r-', 'LineWidth', 2, 'DisplayName',
'Regression Line');
```

```
                title(app.PlotAxes, sprintf('%s: Rainfall vs. Yield (R\xb2=%.4f)', cropName,
rSquared));
                xlabel(app.PlotAxes, 'Rainfall (mm/year)');
                ylabel(app.PlotAxes, 'Yield (hg/ha)');
                legend(app.PlotAxes, 'show', 'Location', 'northwest');
                grid(app.PlotAxes, 'on');
                hold(app.PlotAxes, 'off');

            catch ME
                uialert(app.Fig, ME.message, 'Training Error');
                app.SummaryText.Text = 'Training Failed!';
                app.PredictButton.Enable = 'off';
                cla(app.PlotAxes);
            end
        end

        function PredictYieldButtonPushed(app, ~, ~)
            if isempty(app.CropModelObj)
                uialert(app.Fig, 'Please train a model first.', 'Prediction Error');
                return;
            end

            try
                % 1. Get input values
                rainfall = app.RainfallEdit.Value;
                pesticides = app.PesticidesEdit.Value;
                temp = app.TempEdit.Value;

                if isempty(rainfall) || isempty(pesticides) || isempty(temp)
                     uialert(app.Fig, 'Please enter values for all three inputs.', 'Input
Error');
                     return;
                end

                % 2. Create the input table (X_new)
                X_new = table(rainfall, pesticides, temp, ...
                    'VariableNames', {'average_rain_fall_mm_per_year', 'pesticides_tonnes',
'avg_temp'});

                % 3. Predict Yield
                predictedYield = app.CropModelObj.predictYield(X_new);

                % 4. Update UI
                resultStr = sprintf('%.2f hg/ha', predictedYield);
                app.ResultText.Text = resultStr;

            catch ME
                uialert(app.Fig, ME.message, 'Prediction Error');
                app.ResultText.Text = 'Prediction Failed!';
            end
        end
```

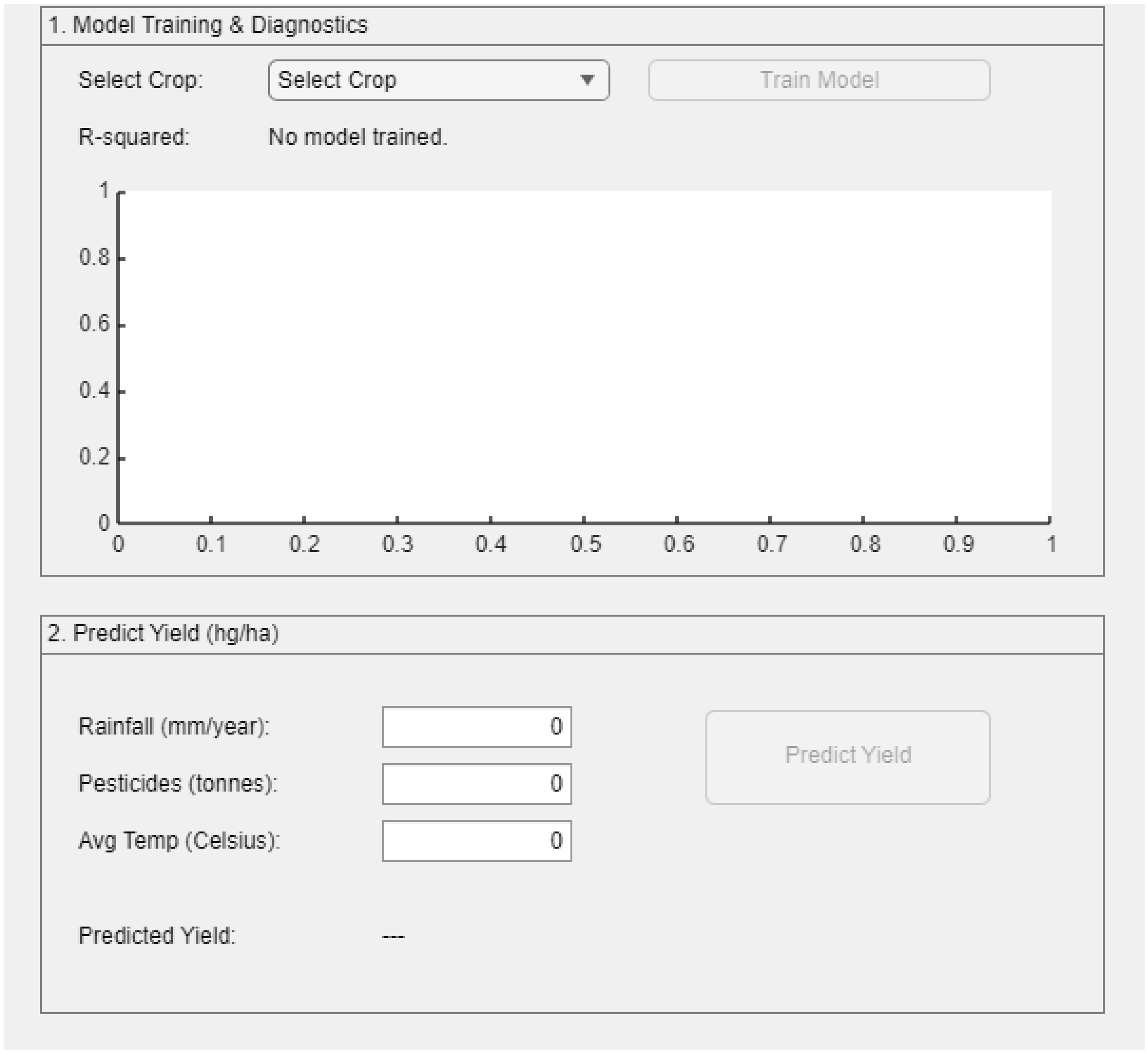
```
    end
end
```

```
Loading data from: yield_df.csv
Warning: Column headers from the file were modified to make them valid MATLAB
identifiers before creating variable names for the table. The original column
headers are saved in the VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as
table variable names.
Preprocessing data...
Data preprocessing complete.

ans = 

  CropYieldEstimatorApp with properties:

               Fig: [1×1 Figure]
      YieldDataObj: [1×1 YieldData]
      CropModelObj: []
      CropDropdown: [1×1 DropDown]
       TrainButton: [1×1 Button]
       SummaryText: [1×1 Label]
          PlotAxes: [1×1 UIAxes]
      RainfallEdit: [1×1 NumericEditField]
    PesticidesEdit: [1×1 NumericEditField]
          TempEdit: [1×1 NumericEditField]
     PredictButton: [1×1 Button]
        ResultText: [1×1 Label]
```


1. Model Training & Diagnostics

Select Crop: Select Crop

Train Model

R-squared: No model trained.

2. Predict Yield (hg/ha)

Rainfall (mm/year): 0

Pesticides (tonnes): 0

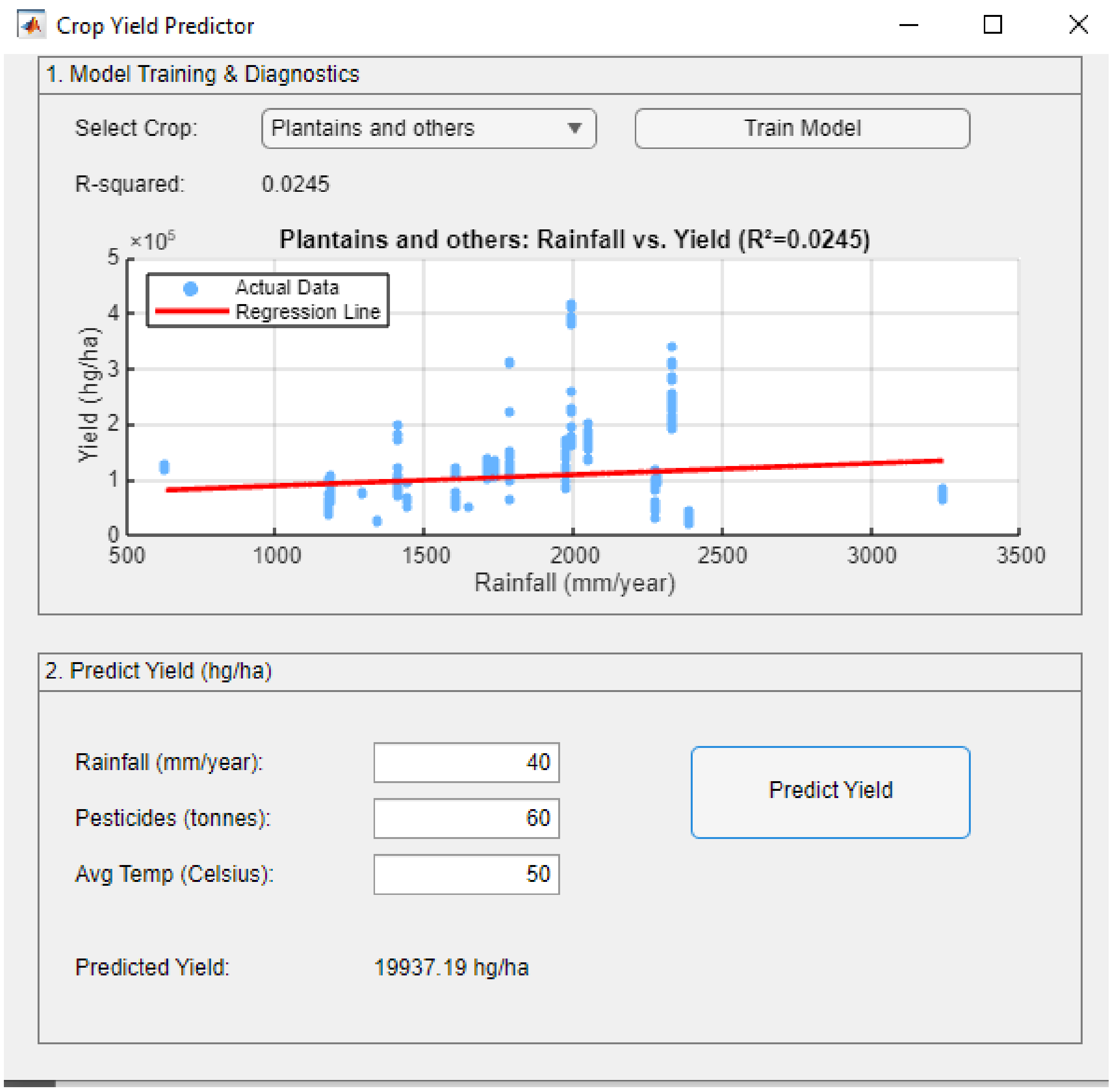
Avg Temp (Celsius): 0

Predict Yield

Predicted Yield: ---

Published with MATLAB® R2024a

Use of a selected module in the of a crop selected within the app for yield estimation.


Crop Yield Predictor

1. Model Training & Diagnostics

Select Crop: Plantains and others

Train Model

R-squared: 0.0245

2. Predict Yield (hg/ha)

Rainfall (mm/year): 40

Pesticides (tonnes): 60

Avg Temp (Celsius): 50

Predict Yield

Predicted Yield: 19937.19 hg/ha

CHAPTER THREE: CONCLUSION AND LEARNING EXPERIENCE

3.1 CONCLUSION

This project successfully implements a complete crop yield estimation system using MATLAB's object-oriented programming capabilities. By combining abstract and concrete classes, the workflow ensures consistency across different crop models and allows easy extension to other algorithms.

The simulation script demonstrates data loading, preprocessing, training, prediction, and visualization, providing actionable insights for agricultural planning.

3.2 LEARNING EXPERIENCE

- OOP Principles: Learned to apply abstraction, encapsulation, inheritance, and polymorphism in MATLAB.
- Data Handling**:** Practiced cleaning, preprocessing, and filtering large datasets safely.
- Modeling & Evaluation**:** Gained hands-on experience with linear regression, interpreting coefficients, R^2, residuals, and visual diagnostics.
- Simulation & Visualization**:** Learned to create time-series plots, scatter plots, and residual plots to assess model performance.
- Software Design**:** Understood the importance of modular, reusable, and maintainable code in scientific workflows.

3.3 References

- Computer programming lecture notes.

- MATLAB Documentation. (2025). *Object-Oriented Programming (OOP) in MATLAB.* MathWorks. Retrieved from https://www.mathworks.com/help/matlab/object-oriented-programming.html