
**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 SIMULATION

It performs a complete crop yield estimation workflow using object-oriented programming principles.

```
% High-level simulation script for Crop Yield Estimation
clear
clc
% 1. Data Initialization and Preprocessing (Encapsulation)
dataManager = YieldData('yield_df.csv');
dataManager = dataManager.preprocessData();

% 2. Select Crop and Get Filtered Data
targetCrop = 'Wheat';
% FIX: Use the public method to filter the data, respecting Encapsulation.
[X_crop, Y_crop] = dataManager.filterDataByCrop(targetCrop);

% Visualizes the raw relationship between features and the target yield
figure('Name', 'Feature Relationships');
FeatureCols = X_crop.Properties.VariableNames;

% Loop through all three features and plot against the yield (Y_crop)
for i = 1:length(FeatureCols)
    subplot(1, 3, i); % 1 row, 3 columns of plots
```

```
    scatter(X_crop{:, i}, Y_crop, 15, 'filled', 'MarkerFaceAlpha', 0.5);

    % Clean up variable names for titles/labels
    readableName = strrep(FeatureCols{i}, '_', ' ');
    title(['Yield vs. ', readableName]);
    xlabel(readableName);
    ylabel('Yield (hg/ha)');
    grid on;
end
sgtitle(['Raw Feature Relationships for ', targetCrop], 'FontSize', 14);

% 3. Model Creation and Training (Polymorphism and Inheritance)
model = LinearRegressionCropModel(targetCrop); % Instantiate a concrete model
model = model.trainModel(X_crop, Y_crop);       % Train the model


% 4. Simulation and Prediction
% Use the last 5 data points for testing/prediction
numTestSamples = 5;
X_test = X_crop(end-numTestSamples+1:end, :);
Y_actual = Y_crop(end-numTestSamples+1:end);

% Predict the yield (Polymorphism ensures the correct method is called)
Y_predicted = model.predictYield(X_test);

% 5. Results and Summary
disp(' ');
disp('  YIELD ESTIMATION SIMULATION RESULTS  ');

model.displaySummary(); % Display model details

disp(' ');
disp(' Prediction vs. Actual for Last 5 Samples ');
Results = table(Y_actual, Y_predicted, 'VariableNames', {'ActualYield',
'PredictedYield'});
disp(Results);

% 1. Original Time Series Plot
figure('Name', 'Time Series Prediction');
plot(1:numTestSamples, Y_actual, 'bo-', 'LineWidth', 2);
hold on;
plot(1:numTestSamples, Y_predicted, 'rx--', 'LineWidth', 2);
title(['Yield Prediction (Time Series) for ', targetCrop]);
xlabel('Sample Index');
ylabel('Yield (hg/ha)');
legend('Actual Yield', 'Predicted Yield');
grid on;
saveas(gcf,'Time series.png');

% 2. Actual vs. Predicted Scatter Plot (New)
figure('Name', 'Actual vs. Predicted Yield');
scatter(Y_actual, Y_predicted, 50, 'filled', 'MarkerFaceAlpha', 0.6);
hold on;

% Plot the y=x line (the line of ideal fit)
```

```
max_val = max([Y_actual; Y_predicted]);
min_val = min([Y_actual; Y_predicted]);
plot([min_val, max_val], [min_val, max_val], 'r--', 'LineWidth', 2);

title(['Actual vs. Predicted Yield for ', targetCrop]);
xlabel('Actual Yield (hg/ha)');
ylabel('Predicted Yield (hg/ha)');
legend('Test Predictions', 'Ideal Fit', 'Location', 'best');
grid on;
saveas(gcf,'y=x line.png');

% 3. Residuals Plot (New)
Residuals = Y_actual - Y_predicted;
figure('Name', 'Residuals Plot');
scatter(Y_predicted, Residuals, 50, 'filled', 'MarkerFaceAlpha', 0.6);
hold on;
saveas(gcf,'Residuals plot.png');

% Plot the y=0 line
plot(xlim, [0 0], 'k--');

title(['Residuals Plot for ', targetCrop]);
xlabel('Predicted Yield (hg/ha)');
ylabel('Residuals (Actual - Predicted)');
grid on;
saveas(gcf,'y=0 line.png');
```

## 2.4 YEILD DATA

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

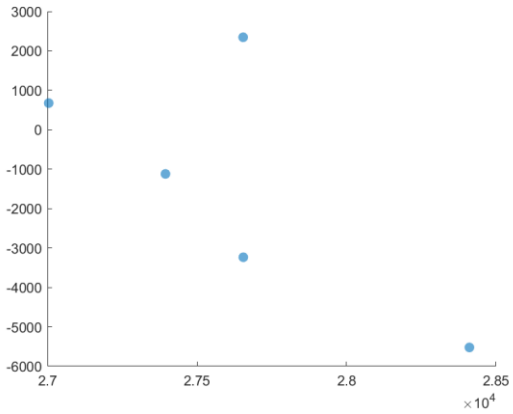
```
classdef YieldData
    % YIELDDATA: Handles the loading, cleaning, and preparation of yield estimation
data.

    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
```

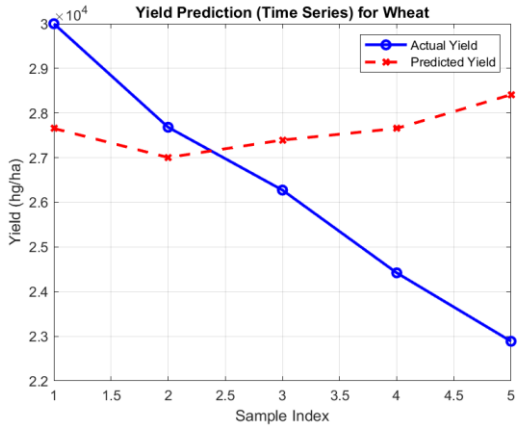
```
    end
end
```

**THE FOLLOWING ARE THE MATLAB PLOTS FROM THE CODES.**

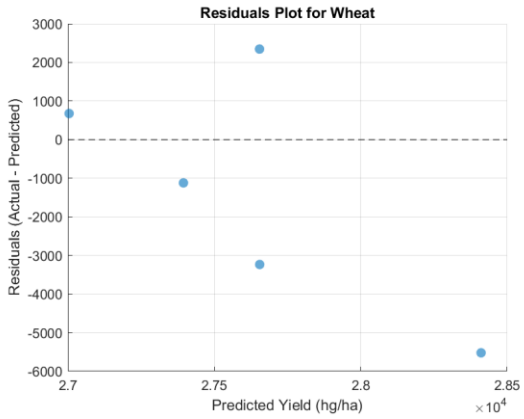
RESIDUAL PLOT



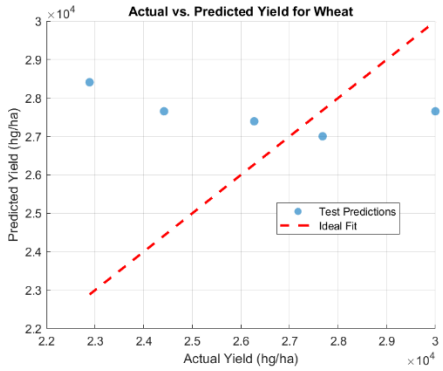

TIME SERIES



Y=0 LINE




Y=X LINE



## 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